Resposta a Incidentes e Análise Forense para Redes Baseadas em Windows 2000

Flávio de Souza Oliveira

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Resposta a Incidentes e Análise Forense para Redes Baseadas em Windows 2000

Flávio de Souza Oliveira[1]

Novembro de 2002

Banca Examinadora:

- Prof. Dr. Célio Cardoso Guimarães (Orientador)
  Instituto de Computação, UNICAMP
- Prof. Dr. Adriano Mauro Cansian
  IBILCE, UNESP
- Prof. Dr. Ricardo Dahab
  Instituto de Computação, UNICAMP
- Profa. Dra. Maria Beatriz F. Toledo (Suplente)
  Instituto de Computação, UNICAMP

1. Financiado por Robert Bosch Ltda.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Oliveira, Flávio de Souza

OL4r Resposta a incidentes e análise forense para redes baseadas em Windows 2000 / Flávio de Souza Oliveira -- Campinas, [S.P. :s.n.], 2002.

Orientador : Célio Cardoso Guimarães
Co-orientador: Paulo Lício de Geus

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computação - Medidas de segurança. 2. Sistemas operacionais. 3. Redes de computadores. I. Guimarães, Célio Cardoso. II. Geus, Paulo Lício de. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Resposta a Incidentes e Análise Forense para Redes Baseadas em Windows 2000

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Flávio de Souza Oliveira e aprovada pela Banca Examinadora.

Campinas, Novembro de 2002

Prof. Dr. Célio Cardoso Guimarães (Orientador)

Prof. Dr. Paulo Lício de Geus (Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aos meus pais, minhas irmãs e a meu irmão...

Agradecimentos

A Deus que esteve sempre comigo durante esta empreitada.

Aos meus pais Zilda e Omar pela confiança, amor e carinho transmitidos a mim durante toda a minha vida.

Às minhas irmãs Adriana e Heloisa pelas palavras de incentivo tão importantes nos momentos difíceis.

Ao meu irmãozinho Paulo Victor pelo carinho, amizade e ternura cativantes.

Ao meu orientador Célio Cardoso Guimarães pela oportunidade, atenção e aprendizado.

Ao meu co-orientador Paulo Lício de Geus pela atenção e aprendizado.

Aos amigos do LAS Alessandro, Cleymone, Diego, Edmar, Fabrício, Jansen, João e Marcelo pela companhia, amizade e discussões enriquecedoras.

À Robert Bosch Ltda. pela ajuda financeira durante o projeto.

A galera do IC pelos churrascos, baladas e demais eventos sociais e esportivos.

Aos professores e funcionários do Instituto de Computação.

Resumo

A preocupação com segurança é requisito essencial para a grande maioria dos serviços e aplicações que utilizam a Internet. Ao contrário do que ocorria nos primórdios da Rede, as organizações que fazem uso da Internet procuram se proteger através da implantação de diversos aparatos como filtros, proxies e VPNs, para evitar o comprometimento de suas informações. Contudo, não há meios de garantir que determinada organização não poderá ser vítima de um incidente de segurança, mesmo que tome todas as precauções para evitar este tipo de acontecimento.

No caso de um incidente de segurança, é vital, mas ainda raro, o estabelecimento de programas de resposta a incidentes, nos quais estão incluídas as metodologias de análise forense.

O presente trabalho apresenta aspectos estruturais deste tipo de programa, além de abordar e discutir metodologias de análise forense para Windows 2000. Baseada nos problemas de tal análise, foi desenvolvida neste trabalho a ferramenta PFSAF (*Pre-Forensic Setup Automation Framework*), com o objetivo de facilitar a implantação de programas de resposta a incidentes, através da automatização de medidas preventivas que possam facilitar uma futura análise forense.

Abstract

Security awareness is an essential requisite for most Internet based services and applications. Organizations that use the Internet, nowadays seek security through several apparatus such as filters, proxies, and VPNs. However, no organization can guarantee to be immune from security incidents, despite all precautions taken to avoid them.

When a security incident occurs, it is vital although infrequent, having been established incident response programs, including forensic analysis methodologies.

This work presents structural aspects for this type of program, besides discussing forensic analysis methodologies for Windows 2000. Based on such analysis we have developed the tool PFSAF(*Pre-Forensic Setup Automation Framework*) with the goal of easing the deployment of incident response programs, through automation of preventive measures that can help a future forensic analysis.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Atualmente, a Internet tornou-se parte do cotidiano de milhões de pessoas ao redor do mundo, informações são trocadas, compras e transações bancárias são efetuadas com uma naturalidade inimaginável até bem pouco tempo atrás. Tal mudança de comportamento, motivou o aparecimento de novos serviços e cada vez mais instituições se predispõem a fazer parte deste movimento. Entretanto, essa corrente fez com que também surgissem crimes procurando explorar este novo tipo de hábito.

Embora hoje as organizações se preocupem em utilizar mecanismos para aumentar a segurança dos sistemas que utilizam a Rede, não há garantias de que elas não poderão ser vítimas de um incidente de segurança, mesmo que sigam todos os tipos de recomendação e implantem as mais modernas tecnologias. A questão é: encontrar meios para que no caso de uma ocorrência deste tipo, a organização possa agir da melhor maneira possível para evitar o agravamento da situação. Para isso existem os programas de resposta a incidente, nos quais geralmente estão incluídas as metodologias de análise forense, e que, ainda hoje, são raros na maioria das organizações.

A preparação para a resposta a um incidente de segurança passa pela elaboração de procedimentos e pela tomada de medidas que possam futuramente auxiliar a realização de uma investigação forense. No entanto existe uma grande escassez de metodologias e estudos abordando este tipo de necessidade em relação às plataformas Windows, apesar de se tratar de um sistema operacional largamente utilizado. Associado a isso, tem-se uma carência de ferramentas de código aberto que facilitem a administração e a implantação de tais medidas na plataforma supracitada.

Neste trabalho desenvolveu-se um estudo de estruturas, ferramentas e técnicas que poderiam constituir boa parte de um programa de resposta a incidentes para organizações baseadas em Windows 2000. O respectivo estudo resultou na implementação de um *framework* que objetiva fornecer, aos times de resposta a incidentes de segurança, um mecanismo para a implantação automatizada de medidas preventivas em redes W2k, o que, por sua vez, pode facilitar uma futura análise forense e ajudar na resolução de crimes eletrônicos envolvendo este tipo de ambiente.

1.1 Organização do Trabalho

No Capítulo 2 são apresentadas algumas estruturas e personagens que podem constituir um programa de resposta a incidentes, suas responsabilidades e respectivas funções. Em virtude do fato de tais eventos estarem ligados ao meio policial, discute-se, no Capítulo 3, toda a problemática legal que pode estar envolvida em uma análise forense computacional, onde ocorre também a apresentação dessa nova disciplina forense e a discussão em torno da necessidade de sua padronização.

Para o entendimento da importância da preparação para uma futura análise, em virtude da complexidade envolvida no processo, tem-se no Capítulo 4 um apanhado de técnicas e ferramentas que podem ser utilizadas durante uma análise forense no W2k, sobretudo durante a chamada *live analysis*. No Capítulo 5 são apresentadas várias ferramentas e estruturas do sistema de arquivos NTFS que podem vir a conter evidências.

Finalmente no Capítulo 6 são destacadas algumas medidas que podem facilitar a análise, seguida da apresentação da ferramenta PFSAF (*Pre-Forensic Setup Automation Framework*), desenvolvida durante este trabalho de mestrado. O PFSAF é composto por um conjunto de scripts desenvolvidos em Perl, que visam fornecer uma maneira centralizada de implantar tais medidas, reduzindo o custo e a complexidade da implantação de programas de resposta a incidentes em organizações baseadas em Windows 2000.

No Apêndice A tem-se um exemplo de SOP (*Standard Operating Procedures*) cuja definição se concentra nos capítulos 2 e 3, já no Apêndice B são apresentados alguns detalhes de implementação do PFSAF acompanhadas dos respectivos trechos de código.

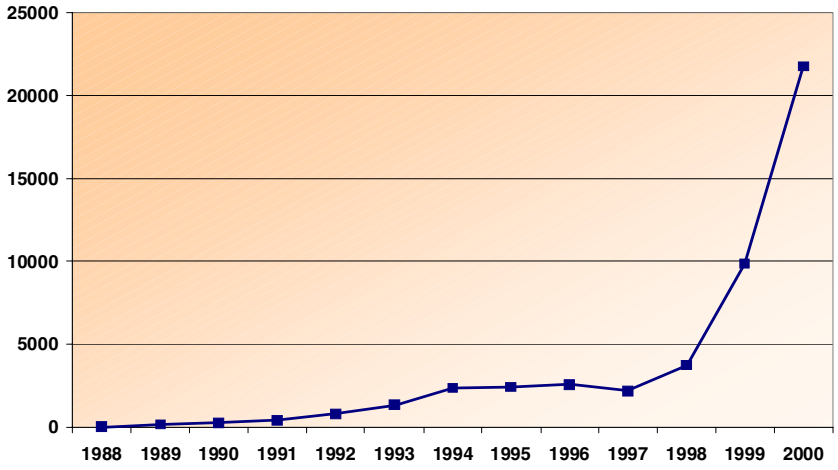
Capítulo 2

Resposta a Incidentes

Em virtude do seu crescimento, atualmente pode-se afirmar que a preocupação com segurança é requisito essencial para a maioria das aplicações em rede. Um bom paralelo foi feito por Jeffrey J. Carpenter, engenheiro de segurança senior do CERT/CC (*Computer Emergency Response Team/ Cordination Center*): "A história da segurança na Internet pode ser comparada à vida em uma cidade. Quando a cidade é pequena, as pessoas se conhecem e confiam umas nas outras, de modo que janelas e portas podem ser deixadas abertas (...). Contudo, quando a cidade cresce, crimes e segurança tornam-se preocupações mais comuns. A Internet pode hoje então ser comparada a uma metrópole, onde as portas e janelas devem permanecer fechadas a maioria do tempo"[3]. Fato esse, que pode ser comprovado pelo aumento do número de ocorrências registradas pelo CERT/CC e exibido na Figura 2.1.

O problema é que mesmo tomando-se todas as medidas necessárias, falhas de segurança podem ocorrer, uma vez que alguma vulnerabilidade ainda não divulgada pode ser explorada ou um novo tipo de ataque pode ser utilizado. Dessa forma, não há como afirmar que um dado aparato de segurança está isento de falhas. Isto se deve principalmente ao fato de que tais aparatos, bem como os serviços oferecidos através da Internet, são compostos por inúmeras peças de software, que por sua vez, possuem milhares de linhas de código que não estão imunes a erros de programação.

Um outro ponto que deve ser considerado é que aproximadamente 70% dos incidentes de segurança possuem origem no interior das organizações [5]. Desta forma, mesmo que a Internet não faça parte de nenhuma etapa dos processos produtivos da instituição, teoricamente ela não está livre de ser vítima de um incidente desse tipo.



Fonte: CERT/CC

Figura 2.1: Crescimento do número de incidentes reportados ao CERT/CC

Tendo em vista que não há esquema de segurança imune a falhas, torna-se então necessária a definição de procedimentos a serem adotados no caso de um ataque bem sucedido, além da presença de pessoal capaz de executar tal função (Seção 2.1.1). No entanto, a preocupação com tal metodologia ainda é muito pequena dentro das organizações conectadas à Rede, mas, por outro lado, a inexistência desses dois personagens pode causar a inviabilização de uma possível ação judicial contra o atacante, além do provável agravamento dos prejuízos financeiros.

Neste capítulo abordar-se-ão os principais aspectos a serem levados em consideração durante a preparação de um programa de resposta a incidentes. O objetivo é deixar o leitor ciente das principais necessidades e implicações de tal programa; no entanto, não é dada ênfase a aspectos técnicos da resposta cuja abordagem será feita nos capítulos 4, 5 e 6. O foco deste capítulo introdutório concentra-se na logística, nas necessidades e nos benefícios de um programa de resposta a incidentes.

2.1 Etapa Pré-incidente

A preparação para a resposta a um incidente de segurança deve começar bem antes do evento propriamente dito, é necessário que já exista toda uma estrutura previamente elaborada para que o acontecimento possa ser tratado com a rapidez e confiabilidade necessárias. Nesta seção serão

discutidos alguns pontos que devem ser observados durante a criação de um programa deste tipo.

São muitas as analogias que podem ser feitas entre a implantação de um programa de resposta a incidentes e a implantação de um plano de contingência contra incêndios [65], como será visto nos parágrafos seguintes. A adoção de procedimentos corretos em ambos os casos, podem muitas vezes minimizar as perdas materiais, além de ainda evitar a perda de vidas humanas ou empregos, neste caso.

Para que uma organização esteja preparada para enfrentar uma catástrofe, tal qual um incêndio, ela deve estar sempre prevenida à espera do acidente. É imprescindível que haja a instalação de extintores, mangueiras e hidrantes, além da colocação de placas indicativas mostrando a localização dos mesmos e das saídas de emergência. Seria também bastante aconselhável a instalação de detectores de fumaça para que os possíveis focos sejam detectados prontamente. Dessa mesma forma, um primeiro aspecto a ser considerado, no caso computacional, é o cuidado com a configuração e monitoramento (i.e. políticas de auditoria) de todos os *hosts* que compõem a rede, uma vez que os incidentes podem possuir desdobramentos totalmente imprevisíveis em virtude do desconhecimento dos objetivos e do nível técnico dos agentes causadores.

A tomada de medidas preventivas durante a configuração e a instalação de programas apropriados, pode minimizar a imprevisibilidade das conseqüências das ações do atacante. Tais medidas tem como objetivo ampliar os vestígios das operações ilegais que porventura possam ser executadas em uma determinada máquina, tornando assim uma posterior análise bem mais fácil de ser efetuada. Dentre elas pode-se citar:

- Configuração minuciosa das políticas de auditoria;
- Utilização de ferramentas que garantam a integridade dos arquivos críticos do sistema;
- Configuração das ACL's (*Access Control List*) e DCL's (*Discretionary Control List*);

Mais detalhes de como efetuar tal configuração preventiva serão vistos no Capítulo 6.

Além da correta configuração de todo o parque de máquinas, a preparação para uma resposta a um incidente pode contemplar uma série de medidas que vão além de parâmetros técnicos de instalação e configuração do sistema operacional. Deve-se considerar por exemplo, a elaboração de uma política de utilização dos recursos de Tecnologia da Informação (TI) que contemple a possibilidade de uma investigação, abordando assim questões como a quebra de privacidade e

monitoramento de atividades. A definição de tal política requer debate entre todos os usuários e discussão da filosofia da organização, o que por muitas vezes não é uma tarefa fácil. Para ajudar nessa etapa, existem diversos modelos disponíveis na Internet, um exemplo seria os que estão presentes no *site* da SANS[1].

Um outro aspecto a ser considerado é o fato de que durante tais acontecimentos não há tempo hábil para que se faça testes ou experiências, já que a pressão para que as atividades retomem a sua normalidade é enorme. Segundo [65] a melhor maneira de se obter a rapidez necessária é através da execução de procedimentos padrões previamente testados (Seção 2.1.2), por pessoal treinado antecipadamente (Seção 2.1.1). Voltando à analogia com o plano de combate a incêndios, durante o isolamento do local do acidente seria bastante razoável que todas as pessoas estivessem cientes da localização das saídas de emergência, bem como dos pontos de encontro. Tal ciência é conseguida invariávelmente através de treinamentos periódicos, para que dessa forma as pessoas possam reagir com a maior naturalidade possível à situação de risco que se apresenta.

2.1.1 Time de Resposta

Um importante passo a ser observado durante a criação da maioria dos programas de combate a incêndios é a criação de uma brigada que será responsável pelo acionamento dos devidos alarmes, coordenação das atividades de combate ao fogo e organização do fluxo de pessoas a fim de evitar o pânico que acabaria por aumentar a gravidade da situação. Da mesma forma, o Time de Resposta (TR) será a entidade responsável por coordenar as atividades antes, durante e depois do incidente de segurança. Apesar de ainda ser muito raro dentro das organizações, esse personagem é de extrema importância para o bom andamento das contramedidas a serem adotadas.

À primeira vista pode parecer inviável a criação de uma equipe deste tipo para a maioria das instituições, nas quais muitas vezes não existem nem mesmo profissionais responsáveis exclusivamente pela segurança dos sistemas. Mas como ver-se-á na Seção 2.1.1.1, o TR não necessita estar fisicamente presente na instituição e nem ser de uso exclusivo desta.

O time de resposta pode ser formado por uma ou mais pessoas com conhecimentos na área de segurança e deve assumir várias responsabilidades, dentre elas:

1. http://www.sans.org/newlook/resources/policies/policies.htm

- Participar da elaboração de uma política de segurança condizente com as ações que possam ser tomadas durante a remediação do incidente;
- Elaborar procedimentos de emergência a serem adotados durante os eventos;
- Preparar simulações de incidentes;
- Coordenar as atividades no caso de um incidente real.

Outro ponto a ser abordado pelo TR é a identificação das prioridades da organização. Do ponto de vista corporativo, um bom tratamento dado a um incidente seria aquele que mais amenizasse o impacto nos negócios da companhia. Os procedimentos devem considerar as prioridades da empresa, bem como saber identificar dentre os possíveis danos quais seriam mais prejudiciais à organização. São exemplos os seguintes itens:

- Comprometimento da reputação da empresa, por exemplo através da desfiguração de páginas web;
- Roubo de propriedade intelectual;
- Modificação ou destruição dos bancos de dados da organização.

Tal identificação serve também para amenizar o choque de objetivos entre o TR, que naturalmente busca remediar o acesso ilegal de forma concomitante à identificação e neutralização do agente causador do incidente, e a empresa que deseja o retorno à normalidade dos negócios o mais rápido possível. O ideal seria o restabelecimento das atividades normais, manipulando as possíveis evidências de forma a garantir sua integridade (e.g. *hash* MD5), não interferindo assim em futura busca ao agente causador do incidente.

2.1.1.1 Interno x Externo

Atualmente, existe uma forte tendência por parte das organizações em terceirizar funções de TI (Tecnologia da Informação), inclusive funções relacionadas à segurança da informação em que geralmente incluem-se funções ligadas ao tratamento de incidentes. Acompanhando esta tendência é cada vez maior o número de empresas que se predispõem a exercer tal função. Contudo, deve-se tomar muito cuidado antes de se opinar entre manter a responsabilidade da manipulação de incidentes como função interna da instituição ou delega-la a uma terceira entidade.

A vantagem de se terceirizar a manipulação de incidentes claramente decorre do custo de se manter pessoal capacitado, enquanto a contratação de uma empresa especializada geraria custo na medida que houvesse demanda de seus serviços. No entanto, um TR interno é dedicado àquela instituição e pode ser acionado rapidamente para enfrentar todos os problemas do dia-a-dia, o que não acontece no caso de um terceiro.

Segundo [65] é muito fácil contratar uma empresa e assumir que se está precavido, enquanto na verdade o contratado não possui disponibilidade para responder a 100% dos eventos. Além disso, segundo o autor, não há nenhuma empresa respeitável que se proponha a cobrir rapidamente todas as requisições de resposta dos clientes.

Os programas de resposta a incidentes mais consistentes, ainda segundo [65], são os que possuem um TR interno robusto, dotado de considerável conhecimento técnico, em associação a um ou mais times externos para o caso de haver demanda de debates técnicos ou mesmo de pessoal. Segundo [65], esta pode ser uma solução extremamente eficaz e ainda assim manter os custos dentro de um limite razoável.

Alguns tipos de TR externo

- TRs Públicos: O primeiro TR público amplamente reconhecido foi o Canegie Mellon CERT, fundado originalmente como uma ramificação do DARPA (*Department of Defense's Advanced Research Projects Agency*), mas que se tratava de um recurso para utilização pública. Atualmente existem dezenas de TRs públicos e um emaranhado de siglas. No Brasil podemos citar:
  - NBSO (*NIC BR Security Office*)[1]: Atendimento geral ao público pertencente ao domínio BR - órgão ligado ao Comitê Gestor;
  - CAIS (Centro de Atendimento a Incidentes de Segurança): Voltado para o atendimento a incidentes ocorridos nos Backbones e Pontos de Presença (PoPs) da RNP (Rede Nacional de Pesquisas);
  - CERT - RS (*Computer Emergency Response Team* - Rio Grande do Sul)[2]: Destinado ao atendimento de clientes conectados à Internet no estado do Rio Grande do Sul;
- TRs Comerciais: Este é o mais recente tipo de TR e sua popularidade vem crescendo continuamente, acompanhando a tendência das empresas em terceirizar serviços relacionados às áreas de TI. TRs comerciais trabalham em base contratual e provêem suporte aos

1. http://www.nic.br/nbso.html
2. http://www.cert-rs.tche.br

seus clientes geralmente apenas quando há demanda. Além da atuação como responsáveis pela manipulação direta ou indireta de incidentes ocorridos nos clientes, estes podem atuar como consultores durante a preparação de políticas e procedimentos, além de efetuar treinamentos, simulações de resposta e testes de invasão, podendo assim, assumir quase que 100% das necessidades das instituições. Entretanto, nenhuma empresa pode garantir o pronto atendimento no caso de haver necessidade de uma manipulação direta de algum evento, isto é, presença física no local do incidente. [65]

- TRs de produtores de Sistemas Operacionais (SO): Muitas empresas produtoras de SO como Sun Microsystems, Microsoft e Hewlett Packard, operam seus próprios TRs com foco em suas plataformas. Diferentemente dos outros TRs citados, estes têm como principais objetivos a documentação, depuração e solução de falhas de segurança que acometem seus sistemas, fornecendo correções para seus clientes e usuários.
  Um detalhe a ser observado é a maneira como as vulnerabilidades são tratadas e apresentadas ao público pelas empresas. Na tentativa de manter sempre a idéia de austeridade e robustez de seus sistemas, muitos dos problemas são apresentadas como correções preventivas ou *"issues"* ao invés de vulnerabilidades propriamente ditas. Este fato se deve ao mercado cada vez mais competitivo, no entanto, quem pode garantir que uma vulnerabilidade não pode ser apenas mascarada até uma próxima versão do sistema, ou simplesmente mantida em sigilo enquanto comercialmente necessário?

- TRs *Ad Hoc*: Tratam-se de profissionais contratados apenas para determinados eventos e que, teoricamente não possuem vínculo com a corporação. São utilizados geralmente quando uma organização não possui um TR interno e nem contrato com nenhuma empresa especializada. Trata-se de último recurso na tentativa de se evitar perdas drásticas e um conseqüente desastre.

A Tabela 2.1 exibe um comparativo entre os TRs supracitados, destacando suas qualidades e deficiências. Note que tais deficiências não indicam pontos a serem corrigidos, mas apenas pontos de interesse das instiuições em geral e que não fazem parte dos objetivos do TR em questão.

Tipo	Qualidades	Deficiências
Públicos	Baixo custo para organização. Boa fonte de estatísticas. Distribuição de alertas variados.	Não pode fornecer atendimento personalizado para cada organização.
Comerciais	Profissionais altamente especializados. Pagamento por demanda.	Desconhecimento da organização interna das instituições. Os problemas diários ainda devem ser manipulados internamente. O tempo para que uma equipe esteja fisicamente no local do incidente pode não ser constante.
SO	Atenções voltadas para a rápida solução das vulnerabilidades encontradas em seus produtos. Contam com a presença de projetistas e programadores dos produtos para resolverem os possíveis problemas.	Foco limitado. Ações podem ser influenciadas por leis de mercado.
Ad Hoc	Usado como último recurso, mas é melhor que a tomada de contramedidas com pessoal não capacitado ou ainda a não tomada de ação alguma. Podem instruir os membros da corporação, sobre as vantagens de se possuir um programa de resposta a incidentes.	Não há vínculos sólidos com a instituição.

Tabela 2.1: Comparativo entre tipos de TR externo.

2.1.2 Definição de Procedimentos

A elaboração e documentação dos procedimentos a serem executados durante um incidente, inclusive referentes à investigação das máquinas afetadas, é essencial para a credibilidade e rapidez da resposta. Como dito anteriormente, não há tempo para experiências, o TR já deve possuir um conjunto de ações pré-definidas mesmo que estas não abordem todas as possíveis situações, uma vez que tal abrangência é difícil de ser conseguida em virtude da quantidade de imprevistos que podem ocorrer. [65]

Na literatura encontramos tais procedimentos definidos como SOP's (*Standard Operating Procedures*). Basicamente estes documentos descrevem como o TR vai prover os serviços de que é encarregado. A idéia dos SOP's tem origem nas instituições militares que seguem rigorosamente inúmeros procedimentos para todas as situações imagináveis. Embora as instituições militares sejam um exemplo extremo, mesmo um TR formado por uma única pessoa deve possuir ao menos os principais passos documentados.

Um aspecto importante a ser levado em consideração durante a elaboração dos SOP's é a atribuição de responsabilidades [65]. Os procedimentos devem considerar aspectos da hierarquia da organização principalmente no que se refere à tomada de decisões, como o contato com TRs ou indivíduos externos, além de instituir uma hierarquia de cargos e responsabilidades dentro do próprio TR: "Quem será responsável pela manipulação das possíveis evidências?", "Quem deve fazer a notificação do ocorrido para outros setores da organização?", entre outros.

No Apêndice A apresenta-se dois exemplos de SOPs que buscam exemplificar as recomendações aqui apresentadas.

2.1.3 Kit de Resposta

Para que o TR seja capaz de fazer a coleta e manipulação das possíveis evidências durante a resposta, precisa já ter selecionado e instalado o grupo de ferramentas que vão auxiliá-lo nesta tarefa. É importante também que todos os envolvidos já estejam familiarizados com a sua utilização.[25]

A ausência de um conjunto de ferramentas condizente com os procedimentos propostos nos SOP's que abordam aspectos técnicos, pode decretar um atraso na resposta, o que conseqüentemente pode resultar em um aumento da extensão dos danos. Caso o TR espere até que uma invasão ou qualquer outro incidente ocorra para identificar e instalar as ferramentas necessárias, não será possível agrupar versões funcionais das peças de software necessárias e nem adquirir o conhecimento necessário para utilizá-las corretamente em tempo hábil, fazendo com que os resultados fornecidos sejam mal interpretados ou imprecisos, inviabilizando uma análise consistente. Tais ferramentas abrangem:

- Ferramentas para captura de dados e configurações: para se efetuar tal tarefa é necessário selecionar ferramentas próprias para análise forense, isto é, que se preocupem em preservar o estado original das evidências, não alterando portanto, tempos de acesso ou qualquer outra informação que venha a deturpar o ambiente;

- Programas para cópias de segurança: com o intuito de preservar a "cena do crime" é interessante que se evite efetuar qualquer tipo de análise sobre as evidências originais (ie. disco rígido), sendo assim recomendável a criação de cópias idênticas para que a partir delas sejam aplicados os devidos procedimentos. No entanto deve ser dada preferência para ferramentas que preservem todos os atributos originais, inclusive espaços vazios. No caso de um disco o ideal é que seja utilizada uma ferramenta para criação de imagens, como o `dd` do Unix, que efetua cópias bit a bit das mais variadas origens;

- Ferramentas para cálculo de *hashes* criptográficos: devem ser utilizadas para garantir a integridade das evidências e dos programas utilizados durante a análise; desta forma é possível provar que não houve qualquer tipo de alteração maliciosa, seja das ferramentas, seja das evidências. Deve-se tomar contudo muito cuidado com o armazenamento dos *hashes* a fim de se evitar qualquer tipo de fraude. Uma medida que poderia ser adotada para se aumentar a segurança do armazenamento é a utilização de extensões HMAC ao invés de *hashes* criptográficos simples [29].

Deve-se atentar ao fato de que quando uma máquina é identificada como alvo de um ataque ou suspeita de uso ilegal por parte de algum agente interno, não é seguro utilizar qualquer tipo de software ou biblioteca presente na máquina. Uma solução muito adotada é a cópia em CD de todos os programas necessários, especializados ou nativos, para neutralizar a tentativa de ocultar informações através da utilização de bibliotecas ou comandos adulterados.

Um outro ponto a ser levado em consideração é a existência de infra-estrutura para a análise, o que pode tratar-se apenas de uma máquina com uma grande capacidade de disco (entenda-se como grande o triplo ou o quádruplo de uma máquina convencional da corporação), onde possam ser copiadas as imagens e os dados das máquinas a serem analisadas, e onde programas e outros artefatos encontrados nas máquinas comprometidas possam ser analisados com segurança.

2.2 Etapa Pós-incidente

Quando um incidente ocorre, é crucial que se evite o pânico para que o plano de resposta previamente elaborado seja seguido com exatidão. Nesta nova etapa também não há uma receita que possa ser seguida integralmente por todo tipo de instituição sem que haja qualquer tipo de adaptação durante o evento. Entretanto, mesmo com as adaptações de última hora, a existência do programa de resposta fornece uma vantagem considerável à organização que está sendo vitimada, em relação a uma que não possui nenhum tipo de preparação [65].

Existem inúmeros procedimentos sendo discutidos e utilizados internacionalmente, entretanto não é possível estabelecer um conjunto de medidas a serem adotadas por todas as organizações [3]. No entanto, podemos destacar alguns que devem ser considerados durante a preparação dos SOP's anteriormente citados (Seção 2.1.2); diversos outros procedimentos específicos ao Windows 2000® serão abordados nos capítulos 4 e 5. Dentre os procedimentos mais importantes pode-se citar:

- Documentação das ações tomadas: todas as ações e decisões tomadas durante a abordagem da máquina vítima devem ser documentadas, sem exceção. O rigor na documentação do processo permite uma futura avaliação da resposta.
- Coleta de informações voláteis: utilizando o CD de ferramentas previamente criado deve-se executar os programas necessários para se coletar o maior número de informações voláteis possível, uma vez que só haverá uma chance de fazê-lo. As evidências colhidas devem ser armazenadas de maneira segura.
- Desligamento da máquina: O desligamento da máquina vítima deve ser feito de forma não convencional para se evitar a possibilidade de execução de algum programa destrutivo. Apenas cortar a fonte de alimentação de energia pode ser uma saída.
- Cópia dos discos: A coleta preliminar de informações (*live analysis*) na maioria das vezes não é suficiente para a solução do evento; por isso deve ser seguida de uma análise posterior (*postmortem*) minuciosa. Os discos devem ser copiados bit a bit (utilizando ferramentas como o *dd*) e seus *hashes* armazenados, para que:
  - Os discos da máquina possam ser liberados, se for o caso;
  - A análise *postmortem* possa ser feita sobre cópias, garantindo assim, que um erro não venha a comprometer o original.

2.2.1 Notificação aos Órgãos Competentes

O ato de reportar um incidente de segurança aos times de resposta públicos é de extrema importância uma vez que eles podem fornecer assistência técnica durante a resposta, além da possibilidade de intermediar o contato com outras organizações vítimas da mesma atividade ilegal. Os incidentes reportados fornecem dados sobre as atividades dos atacantes permitindo o cálculo de estatísticas e elaboração de documentos que visem esclarecer a comunidade sobre as melhores práticas para se evitar incidentes. [11]

Uma das principais missões dos TRs públicos é prover um ponto de contato confiável para tratar de problemas de segurança que envolvam a Internet. Eles facilitam a comunicação entre especialistas que estejam trabalhando na solução de tais problemas, funcionando como centralizador das ações para identificação e correção de vulnerabilidades das aplicações afetadas. Os TRs públicos também podem indicar documentação, fornecer sugestões para recuperação dos sistemas e compartilhar informações sobre as atividades mais recentes dos atacantes.

Em virtude de receberem notificações de ataques provenientes das mais diferentes organizações nas mais diferentes localidades, este tipo de instituição pode correlacionar incidentes que possuam características semelhantes ou envolvam o mesmo atacante; dessa forma podem elaborar estatísticas e levantamentos que podem justificar os investimentos feitos na prevenção de incidentes. Além disso, a experiência adquirida durante a manipulação de incidentes envolvendo os mais diferentes cenários, permite o acumulo de técnicas e documentação que podem ser muito úteis às organizações que estejam enfrentando tais problemas.

Em virtude desses motivos, é bastante recomendável que os procedimentos do programa de resposta a incidentes abordem a necessidade de notificar os TRs públicos, mesmo que não haja necessidade de nenhum tipo de ajuda técnica. Dessa forma, além de se estar respeitando a RFC 1281 [49], que prevê a notificação de incidentes como responsabilidade de todas as organizações conectadas à Rede, estar-se-á contribuindo para a melhoria da segurança na Internet como um todo.

2.3 Resumo

Neste capítulo foram abordados alguns aspectos básicos acerca da criação de programas de resposta a incidentes. Não há, contudo, a pretensão de impor tal modelo a toda e qualquer orga-

nização; a intenção é que ele possa ser usado como ponto de partida para a criação de programas semelhantes, que utilizem os principais requisitos aqui apresentados como matéria prima para futuras adaptações, acréscimos e decréscimos, sempre que necessário.

Capítulo 3

Forense Computacional

No Capítulo 2 foram abordados diversos aspectos relacionados à estrutura de programas de resposta a incidentes de segurança. Tais programas podem englobar diversos processos, desde a elaboração de políticas hierárquicas para acesso a informações privilegiadas, até procedimentos técnicos que lidam com a obtenção de evidências durante a tentativa de solucionar algum crime eletrônico.

A investigação técnica em busca da solução de um crime pode estar fora da alçada do TR, dependendo da gravidade do incidente, do conhecimento e disponibilidade de seus componentes. Cabe ao próprio TR julgar se a resolução de determinado caso está dentro de sua capacidade ou se será necessário acionar uma terceira entidade, como a Polícia Federal por exemplo.

Neste capítulo tratar-se-á de problemas relacionados a essa investigação técnica. O objetivo é expor a problemática legal que pode estar envolvida durante este processo e expor as necessidades relacionadas à padronização de procedimentos. Na Seção 3.3, é apresentado o artigo "*Forense Computacional: Aspectos Legais e de Padronização*" publicado nos anais do I Workshop em Segurança de Sistemas Computacionais (Wseg'2001) [44]. Este tem como foco principal, a necessidade de um esforço internacional de padronização, devido à natureza transjuridicional de diversos crimes cometidos através da Internet [40].

3.1 Ciência Forense

Segundo [59], a ciência forense é aquela exercida em favor da lei para uma justa resolução de um conflito. Em outras palavras ciência forense seria, em sua origem, aquela que se baseia em procedimentos científicos para a obtenção de informações que possam ser úteis durante uma

disputa judicial. Este termo está relacionado ao meio policial e até bem pouco tempo atrás, não tinha qualquer relação com o meio computacional. Entretanto, o aumento do uso cotidiano do computador e da Internet por parte das empresas e usuários domésticos, fez com que surgissem crimes que explorassem esse novo tipo de comportamento. São Crimes praticados por criminosos que apenas aprenderam a utilizar uma nova ferramenta, ou, em sua maioria, por pessoas de conduta aparentemente íntegra no mundo real, mas que buscam se beneficiar do virtual anonimato conferido pela Rede para praticar atos ilícitos.

Para que as agências legais[1] pudessem lidar com este novo tipo de crime e ajudar a justiça a condenar estes criminosos, criou-se a forense computacional. Segundo [41], a forense computacional pode ser definida como sendo a ciência de adquirir, preservar, recuperar e exibir dados que foram eletronicamente processados e armazenados digitalmente.

Assim como ocorre nas outras disciplinas forenses, o processo de análise no meio computacional é metódico e deve seguir procedimentos previamente testados e aceitos pela comunidade científica internacional, de forma que todos os resultados obtidos durante uma análise sejam passíveis de reprodução.

3.2 Manipulação de Evidências

A forma como o TR executa seus procedimentos e investigações durante a resposta a incidentes pode ter implicações durante uma futura ação judicial contra um atacante. Caso o TR não consiga comprovar a autenticidade dos resultados obtidos durante a análise forense, todo o processo legal pode ser comprometido.

Os resultados, referidos no parágrafo anterior, podem ser qualquer tipo de informação que ajudem a responder perguntas relacionadas ao incidente, do tipo: “O que foi feito?”, “Quando foi feito?”, “Quem fez?” e “Como se infiltrou?”. Tais informações são o que podemos chamar de evidências; segundo [28], uma definição simplificada de evidência seria qualquer informação que possua valor probatório, descartando ou confirmando alguma teoria sobre a verdade dos fatos.

Caso o TR opte por efetuar todo o processo de análise, com o intuito de responder às perguntas supracitadas, todos os procedimentos adotados por ele devem ser rigorosamente docu-

1. Polícia, Órgãos de Defesa Civil e Militar

mentados e todas as medidas para a garantia da integridade das informações devem ser tomadas, pois de outra forma as informações obtidas não poderão ser utilizadas em um tribunal.

3.2.1 Padronização

Segundo [30], uma metodologia padrão provê proteção às evidências através da definição de passos comuns que devem ser seguidos durante o processo de investigação. Caso os procedimentos utilizados na manipulação das evidências sigam todas as recomendações previstas nos padrões, não há como contestar sua integridade, fazendo com que possam ser utilizadas como prova em um julgamento.

A importância da existência e da correta aplicação de uma metodologia padrão pôde ser comprovada durante o caso envolvendo o famoso jogador de futebol americano O. J. Simpson. O. J. foi acusado de duplo assassinato após ser apontado como principal suspeito de ter matado sua ex-mulher e um amigo dela. Entretanto, mesmo tendo sido encontradas diversas marcas de sangue, cujo exame de DNA comprovaram pertencer a Simpson, os resultados advindos da análise forense efetuada sobre as amostras de sangue foram atacados pela defesa e não puderam ser utilizados como prova pela promotoria. A estratégia da defesa, neste caso, não foi questionar o resultado dos exames de DNA, cuja possibilidade de erro é quase nula, e sim a metodologia utilizada para manipular as amostras. Este caso teve enorme repercussão nos EUA e forçou as agências legais norte-americanas a reavaliarem grande parte de seus procedimentos [13].

Apesar do consenso geral em torno da necessidade de padronização, a definição de procedimentos para a forense computacional encontra dificuldades, uma vez que existem dezenas de tipos de mídia, sistemas operacionais com inúmeras versões, diferentes tipos de hardware e arquitetura. Além disso, o fato de que cada caso possui circunstâncias praticamente únicas, dificulta sobremaneira a definição de procedimentos que cubram todos os possíveis aspectos de uma análise.

3.3 Artigo

FORENSE COMPUTACIONAL: ASPECTOS LEGAIS E PADRONIZAÇÃO

Célio Cardoso Guimarães
Instituto de Computação - UNICAMP
CP 6176 - 13083-970 Campinas - SP
celio@ic.unicamp.br

Flávio de Souza Oliveira
Instituto de Computação - UNICAMP
13083-970 Campinas - SP
flavio.oliveira@ic.unicamp.br

Marcelo Abdalla dos Reis
Instituto de Computação - UNICAMP
13083-970 Campinas - SP
marcelo.reis@ic.unicamp.br

Paulo Lício de Geus
Instituto de Computação - UNICAMP
CP 6176 - 13083-970 Campinas - SP
paulo@ic.unicamp.br

Resumo

Com o advento da computação e o surgimento da Internet tornaram-se possíveis vários tipos de crimes eletrônicos, o que vem obrigando as agências legais a se prepararem para investigar casos que envolvam a computação. Contudo, em grande parte dos casos, os delitos são transjuridicionais, aumentando assim, a necessidade de intercâmbio e impulsionando a padronização no tratamento de evidências digitais.

3.3.1 Introdução

A forense computacional é um campo de pesquisa relativamente novo no mundo e está se desenvolvendo principalmente pela necessidade das instituições legais atuarem no combate aos crimes eletrônicos. No Brasil conta-se ainda com poucos pesquisadores na área e existem poucas normas estabelecidas, o que gera um grande número de possibilidades de pesquisa.

A eliminação de fronteiras oferecida pela Internet criou um grande problema para as organizações de combate ao crime, uma vez que facilitou em muito a ocorrência de crimes eletrônicos onde a vítima e o criminoso encontram-se em países distintos. Criou-se assim a obrigatoriedade da existência de intercâmbio de informações e evidências eletrônicas entre as agências. Contudo,

por se tratar de uma necessidade muito recente, ainda não se conta com padrões internacionais para o tratamento desse tipo de evidência, sendo que o valor jurídico de uma prova eletrônica manipulada sem padrões devidamente pré-estabelecidos poderia ser contestável.

Este trabalho é um *survey* que aborda basicamente o problema da padronização da análise forense computacional, bem como algumas implicações legais ligadas à sua prática. O objetivo é fornecer ao leitor um panorama do atual estágio do debate e apresentar as principais entidades ligadas ao assunto. Existe contudo, a preocupação de apresentar a ciência forense para aqueles que não estão familiarizados com ela, como se pode constatar na Seção 3.3.2.

3.3.2 Forense Computacional

A Forense Computacional foi criada com o objetivo de suprir as necessidades das instituições legais no que se refere à manipulação das novas formas de evidências eletrônicas. Segundo [41] ela é a ciência que estuda a aquisição, preservação, recuperação e análise de dados que estão em formato eletrônico e armazenados em algum tipo de mídia computacional.

> *"Gathering and analyzing data in a manner as free from distortion or bias as possible to reconstruct data or what has happened in the past on a system."* Dan Farmer e Wietse Venema [19]

Ao contrário das outras disciplinas forenses, que produzem resultados interpretativos, a forense computacional pode produzir informações diretas, que por sua vez, podem ser decisivas em um dado caso [41]. Isso pode ser notado no exemplo que se segue: no caso de um assassinato, o legista verifica que há traços de pele em baixo das unhas da vítima, isso é interpretado como um indício de que houve luta antes da consumação do crime, contudo não passa de uma interpretação. Já no caso de uma perícia em uma máquina suspeita podem ser conseguidos arquivos incriminadores como diários e agendas.

A resolução de um mistério computacional nem sempre é fácil, existe a necessidade de não se observar o sistema como um usuário comum e sim como um detetive que examina a cena de um crime [19]. Felizmente os programadores levam alguma vantagem neste assunto, pois muitas das habilidades necessárias para se procurar um erro em um código fonte, são também necessárias para uma análise forense, tais como: raciocínio lógico, entendimento das relações de causa e efeito em sistemas computacionais e talvez a mais importante, possuir uma mente aberta.[19]

Uma perícia em um computador suspeito de invasão ou mesmo um computador apreendido em alguma batida policial envolve uma série de conhecimentos técnicos e a utilização de ferramentas adequadas para análise. Existe a necessidade de se conhecer minúcias do sistema operacional para que se tenha uma noção global de todos os efeitos das ações do perito [19]. Quanto à necessidade de se utilizar ferramentas específicas para análise, esta decorre da obrigatoriedade de não se perturbar o sistema que está sendo analisado, perturbações essas que podem ser traduzidas como mudanças nos tempos de acesso aos arquivos (MACtimes) por exemplo, anulando assim uma das mais poderosas formas de se reconstituir o que aconteceu na máquina em um passado próximo. Ferramentas convencionais não têm a preocupação de manter a integridade dos tempos de acesso.

3.3.2.1 Privacidade

Ao se fazer uma análise forense em uma máquina, sobretudo se ela atua como servidor de e-mail, arquivos, entre outros, deve-se tomar uma série de cuidados a fim de se evitar a invasão da privacidade dos usuários do sistema. São raras as vezes em que se deve fazer uma busca em todos os arquivos de uma máquina, seja devido a natureza do que se procura ou por limitações de processamento, já que um grande servidor pode conter uma capacidade de armazenamento muito grande, o que pode tornar proibitiva tal operação. O ideal é que se defina um escopo, restringindo ao máximo a área de atuação da análise, evitando-se violar a privacidade de usuários inocentes.

O problema da violação de privacidade muitas vezes pode ser contornado através da instituição de uma política de segurança clara e de conhecimento de todos os usuários, que contemple a possibilidade de vistoria em arquivos, e-mails e outros dados pessoais. Tal possibilidade deve ser seguida pela identificação de quem teria o poder para vasculhar os arquivos alheios, das circunstâncias em que essa medida pode ser tomada e de como o dono dos arquivos será notificado.

Políticas de segurança que contêm tais abordagens são bastante polêmicas, assim como a decisão de se instalar de câmeras de vigilância na sala de café, ou na sala onde se concentram os servidores em uma empresa. Contudo, deve-se conseguir um consenso na definição dos eventos que podem gerar a “quebra do sigilo eletrônico”, de forma que esta possibilidade deve ser tratada como uma medida extrema.

3.3.2.2 Implicações Legais

As evidências resultantes da análise forense podem afetar inúmeras investigações dramaticamente. Segundo [41], nenhuma nova disciplina forense teve tanto potencial desde o DNA.

No Brasil não existem normas específicas que regem a forense computacional, contudo existem normas gerais que abrangem todos os tipos de perícia (ditadas no Código de Processo Penal), podendo ser adotadas no âmbito computacional, salvo algumas peculiaridades. No caso de uma perícia criminal existe a figura do Perito Oficial (dois para cada exame), na qual seu trabalho deve servir para todas as partes interessadas (Polícia, Justiça, Ministério Público, Advogados, entre outros). Para se fazer perícia criminal, o profissional precisa ter nível universitário e prestar concurso público específico, podendo existir porém, a figura do perito *ad hoc* para o caso de não existirem peritos oficiais disponíveis [17].

A responsabilidade do perito no exercício da sua função deve ser dividida em duas partes distintas: aquela do ponto de vista legal, onde lhe são exigidas algumas formalidades e parâmetros para a sua atuação como perito; e as de ordem técnica, necessárias para desenvolver satisfatoriamente os exames técnico-científicos que lhe são inerentes [17].

O perito deve seguir à risca as normas contidas no Código de Processo Penal, dentre elas pode-se destacar duas para exemplificar a sua possível abordagem computacional:

- Art. 170. *Nas perícias de laboratório, os peritos guardarão material suficiente para a eventualidade de nova perícia. Sempre que conveniente, os laudos serão ilustrados com provas fotográficas, ou microfotográficas, desenhos ou esquemas.*

É sempre possível fazer cópias assinadas digitalmente das mídias que estão sendo investigadas para que possam ser feitas análises futuras se necessário. Na verdade o interessante é sempre atuar em cima de cópias, como será visto nos capítulos 4 e 5.

- Art. 171. *Nos crimes cometidos com destruição ou rompimento de obstáculo a substração da coisa, ou por meio de escalada, os peritos, além de descrever os vestígios, indicarão com que instrumentos, por que meios e em que época presumem ter sido o fato praticado.*

Existe a necessidade de se documentar todas as ações dos examinadores, registrando também as evidências encontradas seguidas de seu possível valor probatório. Além disso, é importante que se procure construir uma linha do tempo, retratando o período em que os fatos ocorreram.

Paralelos assim podem ser feitos a fim de se garantir o valor judicial de uma prova eletrônica, enquanto não existir uma padronização das metodologias e nem uma legislação específica regendo a análise forense computacional.

3.3.3 Padronização na Aquisição de Evidências

Um antigo problema encontrado pelas instituições legais norte americanas, era a identificação de recursos dentro da organização que pudessem ser usados para se examinar uma evidência computacional, uma vez que esses recursos estavam espalhados através das agências. Atualmente parece existir uma tendência à mudança desses exames para o ambiente laboratorial. Em 1995, uma pesquisa conduzida pelo serviço secreto norte americano indicou que 48% das agências tinham laboratórios de forense computacional e que 68% das evidências encontradas foram encaminhadas a peritos nesses laboratórios e, segundo o mesmo documento, 70% dessas mesmas agências fizeram seu trabalho sem um manual de procedimentos [43].

Políticas devem ser estabelecidas para a manipulação de uma evidência computacional e, a partir dessas políticas, desenvolver protocolos e procedimentos. Tais políticas devem refletir um consenso da comunidade científica internacional, provendo resultados válidos e reproduzíveis. Contudo, a forense computacional é diferente das outras disciplinas forenses, uma vez que não se pode aplicar exatamente o mesmo método a cada caso [41]. Tome como exemplo a análise feita no DNA recolhido de uma amostra de sangue na cena de um crime, pode-se aplicar exatamente o mesmo protocolo a toda amostra de DNA recebida (elimina-se as impurezas e o reduz à sua forma elementar). Quando se tratam de ambientes computacionais não se pode executar o mesmo procedimento em todos os casos, uma vez que se têm sistemas operacionais diferentes, diferentes mídias e diversas aplicações.[41]

3.3.3.1 Principais Entidades

- IOCE (*International Organization on Computer Evidence*): Principal entidade internacional centralizadora dos esforços de padronização. Ela foi estabelecida em 1995 com o objetivo de facilitar a troca de informações, entre as diversas agências internacionais, sobre a investigação de crimes envolvendo computadores ou outros assuntos relacionados a forense em meio eletrônico. A IOCE identifica e discute assuntos de interesse dos seus constituintes, facilitando assim a disseminação da informação e desenvolvendo reco-

mendações para os membros da organização. Além de formular padrões para evidências computacionais, a IOCE desenvolve serviços de comunicação entre as agências e organiza conferências.

- SWGDE (*Scientific Working Group on Digital Evidence*): Criado em 1998, ele é o representante norte-americano nos esforços de padronização conduzidos pela IOCE;
- HTCIA[1] (*High Technology Crime Investigation Association*): Organização sem fins lucrativos que visa discutir e promover a troca de informações que possam auxiliar no combate ao crime eletrônico;
- IACIS (*International Association of Computer Investigatibe Specialists*): Trata-se de uma associação sem fins lucrativos, composta por voluntários, com o intuito de atuar no treinamento em forense computacional;
- SACC (Seção de Apuração de Crimes por Computador): Atua no âmbito do Instituto Nacional de Criminalística/Polícia Federal, a fim de dar suporte técnico às investigações conduzidas em circunstâncias onde a presença de informação em formato digital é constatada;

3.3.3.2 Padronização Internacional

Com o advento da Internet e da consolidação do mundo globalizado, tornaram-se comuns as notícias de crimes transjuridicionais, obrigando as agências legais de vários países definirem métodos comuns para o tratamento de evidências eletrônicas. Evidentemente cada nação conta com sua legislação e não seria possível a definição de normas gerais para todos. A padronização aqui citada refere-se à troca de evidências entre países.

Atualmente já existem padrões definidos e sendo aplicados de forma experimental. Eles foram desenvolvidos pelo SWGDE e apresentados na *International Hi-Tech Crime and Forensics Conference* (IHCFC), que foi realizada em Londres, de 4 a 7 de outubro de 1999. Os padrões desenvolvidos pelo SWGDE seguem um único princípio, o de que todas as organizações que lidam com a investigação forense devem manter um alto nível de qualidade a fim de assegurar a confiança e a exatidão das evidências. Esse nível de qualidade pode ser atingido através da elaboração de SOPs (*Standard Operating Procedures*), que devem conter os procedimentos para

1. http://www.htcia.org

todo tipo de análise conhecida, e prever a utilização técnicas, equipamentos e materiais largamente aceitos pela comunidade científica (Apêndice A).[57]

3.3.3.3 No Brasil

Ainda não existe padronização em andamento, apenas trabalhos feitos a pedido da polícia federal, trabalhos esses direcionados ao público leigo composto por promotores e juizes federais, além de alguns trabalhos acadêmicos, como pode ser observado em [44] e [50].

Seguem abaixo algumas instituições que possivelmente estariam envolvidas em um esforço de padronização nacional:

- NBSO[1] (*Network Information Center (NIC) - Brazilian Security Office*): atua coordenando as ações e provendo informações para os sites envolvidos em incidentes de segurança;
- CAIS[2] (Centro de Atendimento a Incidentes de Segurança): tem por missão o registro e acompanhamento de problemas de segurança no backbone e PoPs da RNP, incluindo auxílio à identificação de invasões e reparo de danos causados por invasores. Cabe, ainda, ao CAIS a disseminação de informações sobre ações preventivas relativas a segurança de redes;
- GT-S: grupo de trabalho em segurança do comitê gestor da internet brasileira;
- SACC: descrito na Seção 3.3.3.1;

3.3.4 Conclusões

A padronização internacional ainda está distante de ser alcançada devido ao gargalo legal envolvido, visto que cada país conta com sua legislação específica. Além das dificuldades técnicas em se conceber padrões flexíveis que se adaptem às rápidas mudanças tecnológicas.

Em se falando de Brasil, fica clara a atual desorganização em torno do problema. É extremamente danoso que o país fique alheio às discussões internacionais, visto que há o risco de haver incompatibilidades futuras entre a legislação internacional e os interesses nacionais.

1. http://www. nic.br
2. http://www.cais.rnp.br

3.3.5 Glossário (17)

- criminalística: ciência que se utiliza do conhecimento de outras ciências para poder realizar o seu mister, qual seja, o de extrair informações de qualquer vestígio encontrado em local de infração penal, que propiciem a obtenção de conclusões acerca deste fato ocorrido, reconstituindo os gestos do agente da infração e, se possível, identificando-o.
- perícia cível: trata dos conflitos judiciais na área patrimonial e/ou pecuniário.
- perícia criminal: é aquela que trata das infrações penais, onde o Estado assume a defesa do cidadão em nome da sociedade.
- perito: denominação dada aquele profissional que realiza os exames necessários para viabilizar a criminalística, qual seja, todos os exames que envolvem o universo possível em cada situação, para chegar a chamada materialidade do delito, também chamado de prova material ou científica.
- perícia: conjunto de exames realizados no universo da criminalística.

3.4 Resumo

Neste capítulo foram abordados conceitos relativos à forense computacional e a problemática legal envolvida em uma análise. Em virtude da necessidade de validação das evidências digitais, notou-se a importância de haver um esforço de padronização de procedimentos em âmbito nacional como também a criação de padrões internacionais para o intercâmbio entre agências legais, devido, principalmente, ao caráter transjuridicional de vários crimes cometidos através da Internet.

Capítulo 4

Forense em Ambientes Windows

Uma vez compreendida a complexidade legal que pode estar envolvida em uma análise forense e conseqüentemente em uma resposta a incidentes, é necessário que o TR esteja, também, preparado técnicamente para conduzir uma análise forense cujo foco está voltado, principalmente, para a chamada *live analysis*, como ver-se-á na Seção 4.2.

Assim como em um programa de resposta a incidentes, não é possível definir um conjunto de técnicas que se aplique a toda e qualquer análise forense, dado o grande número de situações e necessidades que podem ser encontradas[65]. Neste capítulo serão discutidos alguns procedimentos e ferramentas que podem ser utilizadas por um TR durante a análise forense de ambientes Windows.

Apesar de ser um sistema operacional largamente utilizado, as metodologias de análise forense para plataformas Windows encontram-se em estágio bem menos avançado de desenvolvimento quando comparadas às das plataformas Unix. Isto se deve principalmente ao fato do Windows ser um sistema operacional de código proprietário, no qual é menos comum o desenvolvimento de software livre e metodologias de domínio público, o que acaba por desestimular a criação de projetos de pesquisa abordando esta plataforma.

4.1 Princípios

A análise forense de ambientes Windows segue alguns princípios básicos que são comuns às análises de qualquer plataforma computacional e que foram originalmente herdados de bases da Ciência Forense em geral. Tais princípios estão fundamentados principalmente em métodos para

prover credibilidade aos resultados, fornecendo mecanismos para a verificação da integridade das evidências e da corretude dos procedimentos adotados.

Um importante aspecto a ser ressaltado é a documentação rigorosa de todas as ações tomadas pelo examinador durante a análise. Devido ao stress causado por um incidente de segurança, muitas vezes a documentação das decisões e ações não são devidamente efetuadas, o que pode prejudicar uma futura ação judicial contra os responsáveis, pois pode inviabilizar uma avaliação do tratamento dado às evidências.

Além da correta documentação, pode-se citar mais alguns princípios das demais disciplinas forenses que foram herdados pelo meio computacional:

- Réplicas: durante os exames laboratoriais efetuados no decorrer de análises forenses dos mais variados tipos é sempre recomendável se separar amostras das evidências para que seja possível a repetição dos processos e a conseqüente confirmação dos resultados. A mesma recomendação é adaptada para o meio computacional onde se busca efetuar as análises sempre sobre cópias idênticas das evidências originais, evitando-se que um erro do examinador venha a comprometer as informações, inviabilizando a realização de um novo exame para a validação dos resultados alcançados;

- Garantia de Integridade: deve haver procedimentos previamente determinados que visem garantir a integridade das evidências coletadas. No mundo real as evidências são armazenadas em ambientes cuja a entrada é restrita, são tiradas fotos, minuciosas descrições das peças são escritas com o intuito de verificar sua autenticidade posteriormente. No mundo virtual a autenticidade e a integridade de uma evidência podem ser verificadas através da utilização de algoritmos de *hash* criptográfico como o MD5, SHA-1 e o SHA-2. Além disso é possível armazena-las em mídias para somente leitura, como CD-ROMs;

- Ferramentas Confiáveis: não há como garantir a confiabilidade dos resultados obtidos durante uma análise se os programas utilizados não forem comprovadamente idôneos. O mesmo ocorre no mundo real onde os experimentos de uma análise laboratorial devem ser conduzidos em ambientes controlados e comprovadamente seguros a fim de que os resultados não possam ser contaminados por alguma influência externa;

4.2 *Live Analisys*

Pode-se definir a *live analysis* como sendo aquela efetuada em um sistema vítima de algum incidente de segurança sem que, anteriormente, tenha sido executado qualquer procedimento para

seu desligamento. Esse tipo de análise é extremamente importante para a investigação, uma vez que é a única oportunidade de se coletar uma série de informações voláteis que não estarão mais disponíveis quando o sistema operacional reiniciar, tais como as conexões de rede atualmente ativas e os programas que estão sendo executados no momento.

O grande problema deste tipo de análise é a falta de domínio sobre a máquina analisada, dado que esta pode ainda estar sob a ação do atacante, contendo programas e bibliotecas desenvolvidas para ocultar informações e ludibriar o investigador. Por este motivo, é necessário que a análise seja efetuada a partir de programas e bibliotecas originários de uma mídia confiável (e.g. CD ROM), para que assim o examinador possa ter garantia da integridade dos binários que está utilizando.

Um possível problema com o qual o investigador irá se deparar durante a seleção das ferramentas que constituirão o seu CD de aplicativos (Kit de Resposta), é a necessidade de determinar as dependências de bibliotecas de seus programas. Uma maneira de descobrir quais DLLs um programa necessita para ser executado é através da utilização do *Dependecy Walker* (`depends.exe`). Esta ferramenta vem com o *Resource Kit* do W2k e analisa toda a árvore de dependências de determinado software, podendo exibir inclusive quais funções são utilizadas e exportadas em cada biblioteca.

Outra alternativa é o `listdlls.exe`[1], desenvolvido por Mark Russinovich. Esta ferramenta é capaz de listar todas as bibliotecas que estão sendo utilizadas por determinado processo. Desta forma, é necessário executar a ferramenta a ser analisada para, posteriormente, utilizar o `listdlls.exe` para obter-se a listagem de DLLs necessárias. O listdlls também pode ser útil durante a *live analysis*, na qual pode ser utilizado para investigar algum processo suspeito em uma máquina invadida.

A estratégia de incluir todas as bibliotecas necessárias à execução dos programas a serem utilizados durante a *live analysis* no Kit de Resposta, pode minimizar a utilização de código originário da máquina vítima. Contudo, apenas essa medida não é suficiente para assegurar que nenhuma DLL do sistema suspeito será consultada. Isto porque quando a máquina é abordada, ela está em funcionamento, logo, já possui diversas DLLs carregadas em sua memória. Caso uma das ferramentas a serem utilizadas durante a análise necessite de uma biblioteca que já esteja na

1. http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml.

memória, ela não será carregada novamente, fazendo com que a DLL presente no CD seja ignorada e abrindo uma brecha para a produção de falsos resultados.[2]

A solução mais apropriada para se evitar o acesso a bibliotecas dinâmicas inseguras é eliminar todo e qualquer acesso dinâmico através da compilação estática de todas as ferramentas necessárias para a análise. No entanto, em virtude da cultura comercial da plataforma Windows, poucas ferramentas desenvolvidas para este sistema possuem código aberto, inviabilizando a recompilação da maioria dos programas e conseqüentemente, o uso de programas estáticos.

Outra solução que pode ser adotada seria a limpeza de todas as DLLs presentes na memória durante a análise, contudo a quantidade de distúrbios que esta ação pode causar é imprevisível, uma vez que o W2k não apresenta mecanismos para efetuar esta operação de forma segura. Tal medida pode causar inúmeras GPFs (*General Protection Fault*) nos programas em execução na máquina, prejudicando em muito o andamento da análise forense, uma vez que um dos princípios básicos deste tipo de procedimento é justamente deturpar o sistema analisado o mínimo possível. [19]

A *live analysis* do W2k ainda representa um grande problema para a forense computacional, em virtude dos problemas citados anteriormente.Nas seções subseqüentes são citados diversos pontos que devem ser verificados durante uma *live analysis*, bem como diversas sugestões de ferramentas para auxiliar o processo e que poderão compor o Kit de Resposta definido no Capítulo 2.

4.2.1 Processos

A análise de processos em sistemas Windows é carente de metodologias e ferramentas, apresentando poucos recursos para manipulação deste fundamento. Para se ter um exemplo, note que não há no NT/W2k nenhuma ferramenta textual nativa que possa exibir informações básicas dos processos de uma máquina, tal como o comando `ps` exibe no Unix.

A escassez de ferramentas específicas torna muito difícil a identificação da função de um processo desconhecido sem a análise de seu respectivo binário ou código fonte. A identificação da função do processo desconhecido é importante para decidir que ações devem ser tomadas para sua neutralização, dado que este pode conter funções destrutivas que eliminam evidências.

Outro aplicativo básico do mundo Unix que merece destaque e que não possui equivalente no Windows é o `kill`, que apesar do nome, não é apenas usado para terminar processos. Trata-

se de uma poderosa interface entre o usuário e o escalonador de processos do sistema operacional, pois possibilita o envio de mensagens que viabilizam, por exemplo, o congelamento de um processo suspeito enquanto se determina a melhor abordagem para sua análise (`kill -STOP`).

Felizmente, aplicativos com funções semelhantes ao `ps` e ao `kill` podem ser obtidos nos *Resource Kits* e através da Internet, contudo estão sujeitos à arquitetura do sistema operacional e possuem funcionalidades bem mais restritas, especialmente no tocante ao `kill`. Dentre eles pode-se citar:

- Pulist: ferramenta do NTRK (*NT Resource Kit*) que pode listar os processos de uma máquina remota. Trata-se de um aplicativo bastante simples que lista somente o nome do processo e o seu PID (*Process IDentification*). Caso não seja fornecido o nome da máquina o `pulist` tentará exibir o usuário associado a cada processo;
- Pstat: ferramenta do NTRK que exibe todas as *threads* em execução e seus respectivos estados, agrupadas por seu processo gerador;
- Pslist[1]: ferramenta criada com o objetivo de suprir a falta de um programa como o `ps` do Unix. O pslist pode exibir a listagem de processos de máquinas remotas e possui diversas opções. Trata-se de um aplicativo mais completo que engloba as funcionalidades do `pstat` e `pulist` (Figura 4.1);
- Process Explorer: ferramenta gráfica para análise de processos em máquinas Windows. Sua tela principal é dividida em duas sessões. Na primeira, pode-se observar todos os processos ativos na máquina, incluindo os usuários a que pertencem. Na segunda sessão pode-se exibir tanto os *handles*[2] associados ao processo selecionado na parte superior da janela, quanto as DLLs que ele está utilizando. Também conta com mecanismo de busca nas informações de determinado processo, cujo escopo é definido pelo tipo de visualização escolhida pelo usuário (*handles* ou DLLs) (Figura 4.2);
- Kill: programa nativo que diferentemente do `kill` do Unix, apenas termina processos;
- Pskill[3]: apenas termina processos, contudo, pode fazê-lo remotamente;

1. http://www.sysinternals.com/files/pslist.zip.
2. Estruturas de dados que manipulam arquivos, conexões de rede, entre outros.
3. http://www.sysinternals.com/files/pskill.zip.

```
D:\>pslist.exe \\jabawin2k

PsList v1.2 - Process Information Lister
Copyright (C) 1999-2002 Mark Russinovich
Sysinternals - www.sysinternals.com

Process information for jabawin2k:

Name          Pid Pri Thd  Hnd    Mem     User Time   Kernel Time   Elapsed Time
Idle            0   0   1    0     16   0:00:00.000   1:42:08.902    1:47:54.730
System          8   8  33  173    212   0:00:00.000   0:00:21.120    1:47:54.730
smss          136  11   6   33    336   0:00:00.010   0:00:00.961    1:47:54.730
csrss         164  13   9  322   1584   0:00:00.410   0:00:02.834    1:47:45.677
winlogon      160  13  14  357   1268   0:00:00.150   0:00:02.643    1:47:43.353
services      212   9  33  521   5228   0:00:00.080   0:00:03.304    1:47:41.481
lsass         224   9  14  278   1248   0:00:00.170   0:00:01.211    1:47:41.461
svchost       388   8   6  257   3056   0:00:00.050   0:00:00.370    1:47:35.812
spoolsv       420   8  10  128   3396   0:00:00.010   0:00:00.801    1:47:35.422
netdde        448   8  10   56   1364   0:00:00.010   0:00:00.110    1:47:35.362
svchost       512   8  16  242   5864   0:00:00.030   0:00:02.163    1:47:29.924
navapsvc      536   8   8   70   5412   0:00:00.490   0:00:04.266    1:47:25.377
npssvc        616   8   5   44   1444   0:00:00.010   0:00:00.130    1:47:21.041
NPROTECT      632   8   4   46   2256   0:00:00.040   0:00:00.420    1:47:20.851
regsvc        668   8   4   76   1220   0:00:00.010   0:00:00.380    1:47:20.631
MSTask        688   8   6  146   3236   0:00:00.060   0:00:00.731    1:47:20.410
nopdb         704   8   3   84   2420   0:00:00.010   0:00:00.110    1:47:19.829
VMwareServi   780  13   2   35    948   0:00:00.020   0:00:00.050    1:47:18.878
WinMgmt       808   8   3   94    152   0:00:03.755   0:00:02.223    1:47:18.718
mspmspsv      836   8   2   48   1264   0:00:00.010   0:00:00.050    1:47:17.977
sentnl32      864  13   6  116   3472   0:00:00.010   0:00:00.230    1:47:17.336
alertsvc      112   8  13  154   4480   0:00:00.010   0:00:00.410    1:47:07.622
Explorer      916   8  10  399   5052   0:00:00.560   0:00:45.895    1:46:53.171
VMwareTray   1148   8   1   24   1132   0:00:00.020   0:00:00.070    1:46:44.338
SymTray      1152   8   1   21   1016   0:00:00.010   0:00:00.110    1:46:42.356
qttask       1224   8   1   30    660   0:00:00.010   0:00:00.010    1:46:36.677
evntsvc      1260   8   2   25    104   0:00:00.010   0:00:00.741    1:46:31.660
navapw32     1228   4   1   31   1412   0:00:00.010   0:00:00.180    1:46:29.297
```

Figura 4.1: Exemplo de execução do *pslist.exe*.

- psexec[1]: este aplicativo pode executar processos em máquinas remotas, sem que para isso seja necessária a instalação de qualquer tipo de cliente ou serviço. Ele pode manipular interfaces interativas normalmente permitindo a execução remota do `cmd.exe`, o que resulta em um ambiente semelhante ao oferecido pelo Telnet;

Antes que a máquina vítima possa ser desligada, se for o caso, é importante registrar todos os processos que estão ativos no sistema, em virtude do fato de que esta é a única oportunidade de fazê-lo. É importante também saber diferenciar processos normais do sistema, de outros que podem indicar atividades ilegais. Por exemplo, a constatação de um processo chamado EVN-

1. http://www.sysinternals.com/files/psexec.zip.

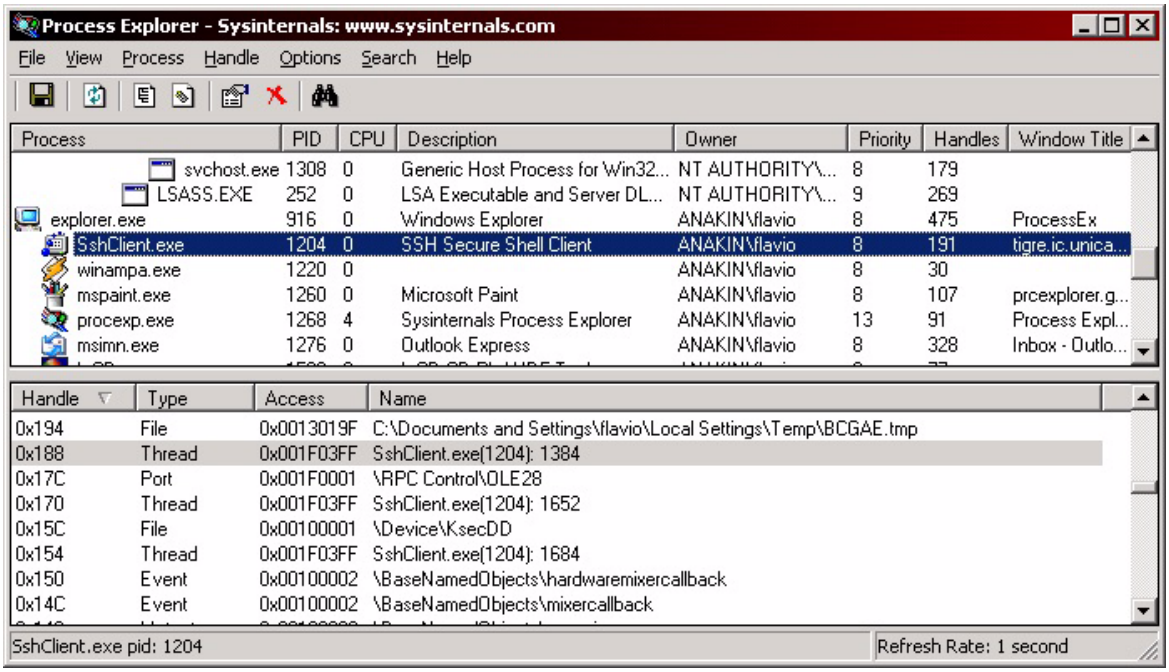


Figura 4.2: Tela principal do Process Explorer.

TVWR sugere que alguém está consultando os logs da máquina. Outro exemplo é o processo USRMGR que pode indicar que alguém está tentando alterar as políticas de auditoria locais, adicionando, excluindo ou alterando contas de usuários. Na Tabela 4.1 temos alguns exemplos de processos de sistema [28].

Processo	Descrição
smss	Session Manager configura o ambiente durante o boot da máquina.
CSRSS	Este é o Client-Server Runtime Server Subsystem, usado para fazer a manutenção do ambiente de sistema Win32 e diversas outras funções vitais.
WINLOGON	Serviço de Logon do Windows.
SERVICES	Gerenciador de Serviços.
LSASS	Local Security Authority Security Service, está sempre sendo executado para realizar autenticações no sistema.
SPOOLSS	Serviço de spool para o subsistema de impressão.
RPCSS	Subsistema para RPC (*Remote Procedure Call*)
ati2plab	Pertencente ao subsistema do driver de vídeo

Tabela 4.1: Alguns processos comumente encontrados no NT/W2k.

Processo	Descrição
EXPLORER.EXE	Responsável pela criação do botão Start, dos objetos do desktop e da barra de tarefas
EVENTVWR	Event Viewer
USRMGR	User Manager
MSDTC	Microsoft Distributed Transaction Coordinator, iniciado automaticamente quando o NT inicia

Tabela 4.1: Alguns processos comumente encontrados no NT/W2k.

4.2.2 Conexões de Rede

Através da análise das conexões de rede e portas TCP/UDP em atividade, pode ser possível construir uma idéia do tipo de utilização que uma determinada máquina está fazendo da rede em um dado momento. Assim como os processos da máquina, esta é a única oportunidade de coletar este tipo de informação volátil que não deixa nenhum tipo de rastro ou histórico depois que a conexão ou atividade é extinta, a não ser que cada aplicação cuide disso através da alimentação de arquivos de *log*, ou seja utilizada alguma ferramenta específica para este fim, como o `TCPWrapper`[1] comumente presente nas distribuições de Linux.

A associação de serviços Internet, como Telnet e HTTP (*Hypertext Transfer Protocol*), com suas respectivas portas padrão, fornece uma idéia do tipo de atividade que gera determinado tipo de conexão, ou seja, a constatação da existência de uma conexão utilizando a porta 80 dá a idéia de que se trata de alguma aplicação fazendo uso do protocolo HTTP, como por exemplo um usuário utilizando um navegador Web.

Note que a interpretação da atividade de rede de uma máquina pode não ser uma tarefa trivial, pois muitas vezes não é possível obter informações suficientes para que possam ser formuladas hipóteses viáveis. Uma conexão utilizando a porta 80, por exemplo, tanto pode ser um navegador Web, quanto um cavalo de Tróia utilizando uma porta que geralmente tem seu tráfego permitido na maioria dos firewalls. Sendo assim, existe a necessidade de interpretação dos dados que estão sendo transferidos, o que muitas vezes é impossibilitada pela utilização de canais seguros de comunicação como SSL (*Secure Sockets Layer*) ou SSH (*Secure Shell*). [12]

Outro ponto que deve ser destacado é a análise da utilização de recursos de rede específicos da plataforma Windows, como o protocolo NetBIOS e os compartilhamentos de recursos através

1. ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz

do protocolo SMB (*Server Message Block*). Abaixo tem-se um conjunto de programas úteis para a coleta de informações relativas a conexões de rede e que podem ajudar a compor o Kit de Resposta:

- Arp: programa nativo que é capaz de exibir o cache de respostas obtidas pelo protocolo ARP. Através deste aplicativo é possível enumerar todos os endereços de enlace com os quais a máquina tem se comunicado no último minuto;
- Netstat: o objetivo desta ferramenta nativa é fornecer estatísticas de utilização da rede, mais especificamente dados sobre os protocolos IP, UDP e TCP. Ele exibe informações que podem ser úteis para o examinador, tais como a lista de conexões ativas na máquina, as portas que estão aceitando conexões da rede e qual o estado atual da tabela de roteamento;
- Nbtstat: ferramenta nativa que exibe informações relacionadas ao protocolo NetBIOS. Através deste aplicativo, o examinador pode consultar o cache de nomes resolvidos através do protocolo NetBIOS, contudo, estas informações são bastante voláteis ficando armazenadas por apenas 10 minutos (Figura 4.4).;
- Tracert: semelhante ao traceroute do Unix, indica quais roteadores estão entre a máquina local e um determinado destino em uma rede. Este aplicativo é nativo e pode ser útil caso sejam observadas rotas de rede suspeitas durante a utilização do `netstat`;
- Fport[1]: trata-se de uma ferramenta extremamente útil durante a análise de portas e conexões suspeitas. Este aplicativo é parte integrante do Foundstone Forensic Kit e indica quais processos estão utilizando quais portas TCP/UDP, além disso, fornece o caminho para seu binário. A idéia é semelhante à do `lsof` do Linux, que pode relacionar processos com operações de escrita e leitura em qualquer tipo de estrutura, o que inclui sockets TCP/UDP. O `fport` é bem mais simples que `lsof`, mas pode ser de grande valia (Figura 4.5);
- Windump[2]: ferramenta para captura de pacotes de rede, pode ser utilizada para armazenar o tráfego de determinada conexão para uma posterior análise. A interpretação de seus dados pode ser muito trabalhosa, devido à grande quantidade de informações que são

1. http://www.foundstone.com/knowledge/zips/FPortNG.zip
2. http://windump.polito.it/install/bin/alpha/WinDump.exe

```
C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:7              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:9              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:17             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:19             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:515            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3002           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3006           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3295           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3299           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3312           0.0.0.0:0              LISTENING
  TCP    10.1.1.44:139          0.0.0.0:0              LISTENING
  TCP    10.1.1.44:3028         0.0.0.0:0              LISTENING
  TCP    10.1.1.44:3028         10.1.1.3:139           ESTABLISHED
  TCP    10.1.1.44:3214         143.106.7.16:22        TIME_WAIT
  TCP    10.1.1.44:3295         143.106.60.15:8080     CLOSE_WAIT
  TCP    10.1.1.44:3299         143.106.60.15:8080     CLOSE_WAIT
  TCP    10.1.1.44:3312         143.106.60.15:8080     ESTABLISHED
  UDP    0.0.0.0:7              *:*
  UDP    0.0.0.0:9              *:*
  UDP    0.0.0.0:13             *:*
  UDP    0.0.0.0:17             *:*
  UDP    0.0.0.0:19             *:*
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1645           *:*
  UDP    0.0.0.0:1646           *:*
  UDP    0.0.0.0:1812           *:*
  UDP    0.0.0.0:1813           *:*
  UDP    0.0.0.0:3003           *:*
  UDP    10.1.1.44:137          *:*
  UDP    10.1.1.44:138          *:*
  UDP    127.0.0.1:3004         *:*
  UDP    127.0.0.1:3005         *:*
  UDP    127.0.0.1:3215         *:*
```

Figura 4.3: Saída do comando netstat.

```
C:\>nbtstat -c

Local Area Connection:
Node IpAddress: [10.1.1.44] Scope Id: []

                    NetBIOS Remote Cache Name Table

        Name               Type       Host Address    Life [sec]
    --------------------------------------------------------------
    JABA            <20>  UNIQUE          10.1.1.45           602
```

Figura 4.4: Saída do comando nbtstat.

```
C:\Temp\Imagens\Visual>fport.exe
FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
700   tcpsvcs        ->  7     TCP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  9     TCP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  13    TCP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  17    TCP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  19    TCP   C:\WINNT\System32\tcpsvcs.exe
556   svchost        ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System         ->  139   TCP
8     System         ->  445   TCP
700   tcpsvcs        ->  515   TCP   C:\WINNT\System32\tcpsvcs.exe
848   MSTask         ->  1025  TCP   C:\WINNT\system32\MSTask.exe
712   ntfrs          ->  3002  TCP   C:\WINNT\system32\ntfrs.exe
1044  inetinfo       ->  3006  TCP   C:\WINNT\System32\inetsrv\inetinfo.exe

700   tcpsvcs        ->  7     UDP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  9     UDP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  13    UDP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  17    UDP   C:\WINNT\System32\tcpsvcs.exe
700   tcpsvcs        ->  19    UDP   C:\WINNT\System32\tcpsvcs.exe
556   svchost        ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System         ->  137   UDP
8     System         ->  138   UDP
8     System         ->  445   UDP
640   svchost        ->  1645  UDP   C:\WINNT\System32\svchost.exe
640   svchost        ->  1646  UDP   C:\WINNT\System32\svchost.exe
640   svchost        ->  1812  UDP   C:\WINNT\System32\svchost.exe
640   svchost        ->  1813  UDP   C:\WINNT\System32\svchost.exe
236   services       ->  3003  UDP   C:\WINNT\system32\services.exe
640   svchost        ->  3004  UDP   C:\WINNT\System32\svchost.exe
640   svchost        ->  3005  UDP   C:\WINNT\System32\svchost.exe
```

Figura 4.5: Saída do comando fport.

geradas, contudo, é possível o armazenamento de seus dados para que eles possam ser analisados posteriormente por programas de interface mais amigável como o `ethereal`[1];

- Net: programa nativo que agrega inúmeras funcionalidades relacionadas à administração de redes Windows. Este aplicativo pode ser útil durante a *live analysis* para que o examinador obtenha dados relativos aos compartilhamentos mapeados (`net use`), compartilhamentos exportados (`net share`) e a lista das sessões em atividade na máquina (`net session`);

1. http://www.ethereal.com/distribution/win32/ethereal-setup-0.9.6.exe.

- Rmtshare: este aplicativo do NTRK tem praticamente as mesmas funcionalidades no `net share`, contudo pode realizar todas as suas tarefas remotamente;
- Rasautou: máquinas com conexões *dialup* podem ser configuradas para efetuarem discagem toda vez que uma aplicação solicite acesso à Internet (*dial-on-demand*) e é comum que algumas aplicações tentem utilizar este recurso por *default*. O Windows 2000 mantém um histórico de todos os endereços IP que foram conectados via *dial-on-demand*, que podem ser consultados através desta ferramenta nativa (`rasautou -s`);

4.2.3 Usuários

A maioria das informações relativas a usuários podem ser analisadas *offline*, contudo, dados como quem está logado na máquina e que processos estão executando podem ser perdidos. Estas informações são importantes, principalmente se não há políticas de auditoria habilitadas para registrarem tais eventos. Consultas no log do sistema são bem mais efetivas neste tipo de análise, entretanto, não havendo logs a serem consultados esta pode ser a única oportunidade de descobrir alguma anomalia na utilização de contas de usuários.

Uma anomalia na utilização de contas poderia ser caracterizada através da constatação de usuários logados em períodos e máquinas não habituais ou a utilização de privilégios administrativos incomuns, por exemplo. Seguem algumas ferramentas relacionadas a este tipo de preocupação:

- Pwdump[1]: aplicativo capaz de copiar os *hashes* criptográficos das senhas dos usuários de um sistema Windows NT/2000, para que posteriormente possam ser submetidos a uma tentativa de quebra de senha utilizando *cracks* como `L0phtcrack`[2] ou `John the Ripper`[3]. Este procedimento pode ser útil caso se esteja lidando com um usuário que não deseja cooperar com as investigações. Note que no caso de uma clonagem do sistema vítima o examinador geralmente deseja se autenticar na máquina como administrador do sistema [28];
- net: Através do comando `net user` é possível consultar quais usuários estão cadastrados na máquina local, além de viabilizar a alteração de seus respectivos dados;

1. http://www.polivec.com/Downloads/pwdump3v2.zip ou http://razor.bindview.com/tools/files/pwdump2.zip
2. http://www.atstake.com/research/lc/application/lc4setup.exe
3. http://www.openwall.com/john/john-16w.zip

- Psloggedon[1]: aplicativo que informa quais usuários estão logados em uma máquina. Esta tarefa pode ser executada tanto local quanto remotamente. Caso o nome do usuário seja informado, este aplicativo tentará localiza-lo em todas as máquinas do domínio;
- Rasusers: ferramenta do NTRK capaz de listar os usuários que possuem permissões para se conectarem, via *dial-up*, em determinada máquina.

4.2.4 Logs

Os logs são a melhor fonte de informação sobre o passado do sistema e a análise de seus dados pode ser a diferença entre o sucesso e o fracasso na resolução de um incidente.

O Windows NT/2000 possue três tipos de log: o *System log*, *Application log* e S*ecurity log*, através dos quais é possível obter informações do tipo [28]:

- Determinar quais usuários têm acessado determinados arquivos;
- Determinar quais usuários têm feito logon no sistema;
- Determinar falhas no logon de usuários;
- Monitorar o uso de determinadas aplicações;
- Monitorar mudanças nas permissões de usuários;

O *System log* é responsável por registrar eventos relativos aos drivers de dispositivos e aos processos gerados pelo sistema operacional. Os eventos normalmente auditados por este log incluem: falhas na inicialização de drivers e no hardware da máquina, endereços IPs duplicados, além da inicialização, pausa e paralisação de serviços [28].

Programas comerciais, que não são nativos do sistema, registram eventos no *Application log*. Tais registros incluem erros e informações que a aplicação deseja registrar. Este log também pode conter eventos auditados pelo *Performance Monitor*, tais como número de logons falhos, quantidade de disco em uso e outras métricas [28].

Qualquer usuário tem permissão para visualizar os registros contidos no *Application log* e no *System log*, o que não ocorre com o *Security log* que registra eventos configurados na política de auditoria (*audit policy*) e que podem ser consultados apenas pelos administradores do sistema. Este log é o mais útil durante a análise forense, contudo não é habilitado por *default*.

1. http://www.sysinternals.com/files/PsLoggedOn.zip.

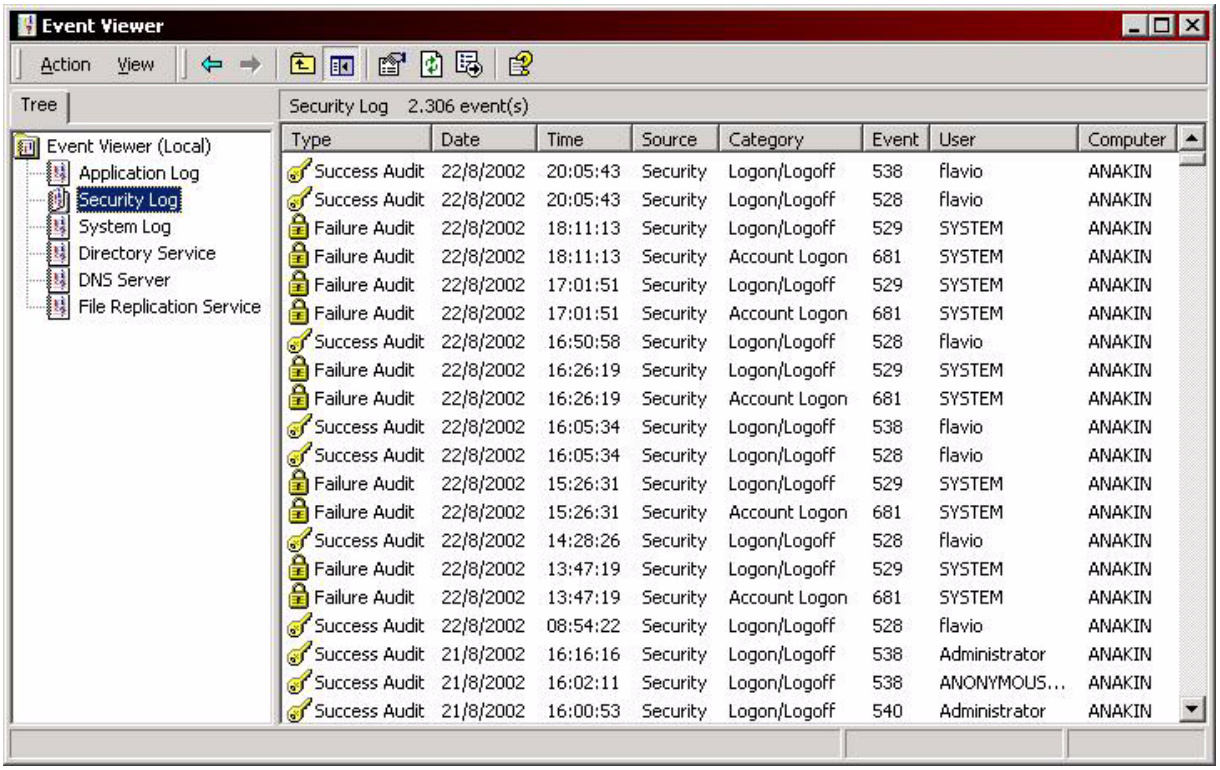


Figura 4.6: Event Viewer.

Na Figura 4.6 tem-se o console Event Viewer do Windows 2000, responsável por exibir os dados referentes a todos os tipos de log citados até o momento. A coluna de maior interesse para o examinador é a *Event,* que possui os códigos[1] dos eventos a partir dos quais é possível construir filtros para eventos específicos. Na Tabela 4.2 tem-se alguns exemplos de códigos com suas respectivas descrições.

Código	Descrição
516	Registros de eventos auditáveis foram descartados
517	Limpeza de log
528	Logon efetuado com sucesso

Tabela 4.2: Alguns códigos de eventos de segurança.

1. Uma lista completa dos eventos de segurança e seus respectivos códigos pode ser obtida em: http://www.microsoft.com/windows2000/techinfo/reskit/ErrorandEventMessages/default.asp.

Código	Descrição
529	Falha no logon
531	Falha no logon seguido de bloqueio de usuário
538	Logoff efetuado com sucesso
576	Uso e associação de privilégios
578	Uso de serviço privilegiado
592	Novo processo criado
593	Processo terminado
595	Acesso indireto a um objeto
608	Mudança na política de permissões
610	Novo relação de confiança entre domínios
612	Mudança na política de auditoria
624	Novo usuário
626	Conta de usuário habilitada
630	Conta de usuário excluída
636	Conta de grupo alterada
642	Alteração em conta de usuário
643	Mudança na política de domínio

Tabela 4.2: Alguns códigos de eventos de segurança.

Abaixo tem-se algumas ferramentas para auxiliar a análise dos logs:

- Ntlast[1]: linha de comando que trata eventos relacionados ao logon/logoff de usuários. A utilização dessa ferramenta pode agilizar a consulta neste tipo de evento podendo fornecer respostas rápidas ao examinador. É possível efetuar buscas por usuários, filtrar eventos relacionados ao IIS (*Internet Information Service*) e ao SMB [21];

- Dumpel: ferramenta que pode ser útil no caso de uma análise dos logs do sistema em uma segunda máquina. As mensagens exibidas nas janelas de descrição dos eventos (Figura 4.9) são muitas vezes construídas no momento da exibição, através da consulta das mensagens em bibliotecas dinâmicas do programa que gerou o evento. Entretanto, quando necessita-se analisar tais logs em uma outra máquina, as DLLs podem não estar

1. http://www.foundstone.com/knowledge/termsofuse.html?filename=ntlast30.zip

```
C:\Temp\Imagens\Visual>NTLast.exe -f -i
flavio          ANAKIN              ANAKIN          Mon Jun 24 05:14:17pm 2002
jansen          ANAKIN              ANAKIN          Fri May 31 08:27:16pm 2002
Administrator   ANAKIN              ANAKIN          Mon May 13 01:23:21pm 2002
Administrator   ANAKIN              ANAKIN          Mon May 13 12:21:25pm 2002
fulano          ANAKIN              ANAKIN          Thu May 09 06:10:31pm 2002
Administration  ANAKIN              ANAKIN          Mon Apr 29 09:30:40pm 2002
Administration  ANAKIN              ANAKIN          Mon Apr 29 09:30:37pm 2002
Administration  ANAKIN              ANAKIN          Mon Apr 29 09:30:31pm 2002
Administration  ANAKIN              ANAKIN          Mon Apr 29 09:30:25pm 2002
Administration  ANAKIN              ANAKIN          Mon Apr 29 09:30:22pm 2002
```

Figura 4.7: Saída produzida pelo NTLast (últimas falhas (-f) em logon interativo (-i))

disponíveis impossibilitando assim a visualização da descrição de alguns eventos (Figura 4.9). Este programa do NTRK gera um arquivo texto contendo todos os eventos e suas respectivas descrições, permitindo a consulta *offline* das mensagens.

```
C:\Temp\Imagens\Visual>DUMPEL.EXE -l security -t -f a:\security.log
Dump successfully completed.
```

Figura 4.8: Exemplo de utilização do dumpel.exe.

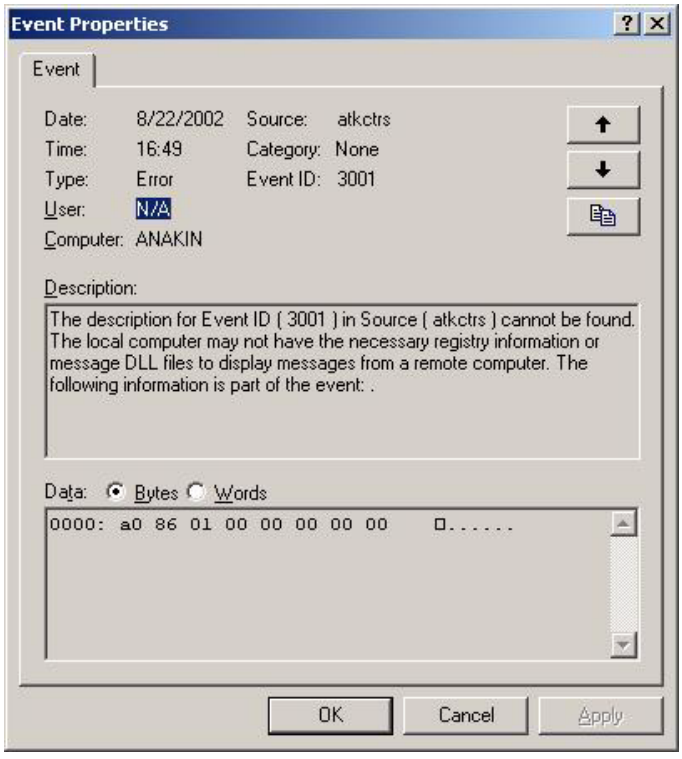


Figura 4.9: Descrição de evento não exibida devido à falta de DLLs.

- Auditpol: programa do NTRK capaz de habilitar e consultar políticas de auditoria remotamente. Tais políticas, como já mencionado nesta seção, não são habilitadas por *default*, acarretando a não geração de logs de segurança. Esta ferramenta também é útil para que o examinador documente em suas anotações quais eventos estão sendo auditados na máquina vítima.

4.2.5 Registro

O registro do Windows é composto por um conjunto de arquivos que contém informações vitais sobre a configuração da máquina, tais como a descrição do hardware, dos programas instalados e de diversos outros componentes do sistema.

O registro pode ser visto como um grande arquivo de log, podendo conter informações importantes para o investigador como quais programas foram instalados na máquina no passado, configurações relacionadas à segurança, traços deixados por cavalos de Tróia e outros programas executados automaticamente durante o *logon* dos usuários (Capítulo 6), além das listas de arquivos recentemente utilizados por diversas aplicações [28].

O registro do Windows 2000/NT é composto de cinco hierarquias ou colmeias (*hives*): HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_CONFIG. Os dados contidos nessas hierarquias são obtidos, em sua maioria, de quatro arquivos de sistema: SAM, SECURITY, SOFTWARE e SYSTEM, armazenados no diretório `%systemroot%\system32\Config`.

Análises no registro podem ser úteis, principalmente, quando há a suspeita de que algum software indevido esteve instalado na máquina vítima. Não é raro que ocorra desinstalação manual de software, durante incidentes de segurança o que pode deixar muitos rastros no registro. Mesmo quando a desinstalação é feita automaticamente, é comum que os programas mantenham as preferências de seus usuários cadastradas no registro.

Abaixo temos algumas ferramentas relacionadas à manipulação do registro:

- Reg: ferramenta textual do NTRK que oferece uma interface completa para manipulação do registro. Através de sua utilização é possível fazer consultas, adicionar e alterar valores entre outras funcionalidades (Figura 4.10), sendo que as hierarquias HKEY_LOCAL_MACHINE e HKEY_USERS podem ser manipuladas remotamente.;

```
G:\Pesquisa\NTRK>REG QUERY HKLM\Software\Microsoft\Windows\CurrentVersion\Run \\JABA
Connecting to remote machine \\JABA

Listing of [Software\Microsoft\Windows\CurrentVersion\Run]

REG_SZ          Synchronization Manager mobsync.exe /logon
REG_SZ          RealTray        C:\Program Files\Real\RealPlayer\RealPlay.exe SYSTEMBOOTHIDE-
PLAYER
REG_SZ          NPS Event Checker       C:\PROGRA~1\NORTON~1\NORTON~3\npscheck.exe
REG_SZ           SymTray - Norton SystemWorks    C:\Program Files\Common Files\Symantec Sha-
red\SymTray.exe "Norton SystemWorks"
REG_SZ          QuickTime Task  C:\WINNT\System32\qttask.exe
[OptionalComponents]
```

Figura 4.10: Consulta remota realizada com o aplicativo reg.exe

- Regdump: o objetivo deste programa do NTRK é gerar arquivos textuais com as informações contidas no registro local ou de máquinas remotas. Ele gera arquivos no formato utilizado pelo Regini, sendo possível importar seu conteúdo novamente para o registro, funcionando exatamente como a opção *export registry* do `regedit`;
- Regedit: o regedit é o editor de registro padrão do Windows 2000, através dele é possível realizar todas as operações efetuadas pelas duas ferramentas apresentadas anteriormente de forma gráfica, também é possível realizar buscas e conexões em registros remotos;
- Regedt32: editor de registro nativo do Windows 2000 que pode representar uma alternativa à sua utilização do regedit, uma vez que apresenta recursos extras tal como a manipulação das permissões associadas a uma determinada chave;

4.2.6 Cópia da memória

Dependendo do caso, a obtenção de uma cópia dos dados presentes na memória do sistema pode ser importante, na medida que esta pode conter senhas, textos em claro de mensagens cifradas, ou o conteúdo de arquivos acessados recentemente [28]. Entretanto, a análise do conteúdo da memória pode ser bastante trabalhosa, pois os dados podem estar incompletos e difíceis de filtrar.

O problema é que as plataformas Windows NT/W2k não apresentam mecanismos para duplicação do conteúdo da memória de maneira condizente com as necessidades de uma análise forense, uma vez que é necessário que a máquina seja reiniciada para que ocorra a cópia. Além disso, a máquina já deve estar previamente configurada para a geração da cópia, pois tal configuração exige que a máquina seja reiniciada uma vez, o que implicaria na perda dos dados que

se deseja armazenar. Informações sobre como configurar e utilizar este mecanismo podem ser obtidos em [37] e [39]. A análise das cadeias de caracteres contidas no arquivo binário de memória pode ser feita através do programa `strings` que é detalhado no Capítulo 5.

4.2.7 Outras Considerações

Existem dois tipos de dado que na maioria dos casos podem ser obtidos com sucesso durante uma análise *postmortem*, que seria aquela efetuada após o deligamento do sistema vitima (Seção 4.3), entretanto, geralmente não há nada que impeça sua coleta durante a *live anlaysis*. Como se voltará a discutir no Capítulo 6, atualmente está se tornando cada vez mais raro a realização de análises *postmortem*, devido, principalmente, a restrições em relação a paralisação de serviços da organização vítima do incidente de segurança.

A primeiro tipo de dado é a coleta de todos os registros de tempo dos arquivos presentes na máquina, os MACtimes (*Modification*, *Access* e *Creation Times*). Através da análise destes dados é possível cruzar informações temporais obtidas de outras fontes de evidências e descobrir quais arquivos foram acessados enquanto o incidente estava ocorrendo. Mais detalhes sobre os MACtimes serão abordados no Capítulo 5. Na Figura 4.11 tem-se um exemplo de utilização do comando `dir` para obter os tempos de modificação dos arquivos da pasta `%systemroot%\System32\config`. Note que a presença de mecanismos de sincronização de relógios na rede tem influência decisiva na análise neste tipo de dado.

O segundo ponto é a documentação das tarefas agendadas no *Task Scheduler*, em virtude da possibilidade de um atacante poder agendar processos maliciosos para serem executados em horários mais convenientes às suas atividades. Os programas agendados podem ser visualizados na pasta `%systemroot%\Tasks` (Figura 4.12) e através do programa nativo `at`, no entanto o comando `at` só é capaz de exibir tarefas agendadas através dele próprio.

4.3 Análise *Postmortem*

Após a realização de uma coleta inicial de informações voláteis, pode ser necessário *postmortem* detalhada, tal análise pode ser descrita como aquela conduzida a partir de cópias das evidências originais em uma máquina preparada para esta tarefa (*Forensic Station*), isto é, contendo ferramentas, boa capacidade de disco e contando com os sistemas operacionais apropriados. Mais detalhes sobre este tipo de análise serão abordados no Capítulo 5.

```
D:\>DIR /T:w /A /S /O:d C:\WINNT\System32\config
 Volume in drive C is 2000
 Volume Serial Number is 40D3-2B28

 Directory of C:\WINNT\System32\config

04/03/2002  10:22                 0 TempKey.LOG
04/03/2002  10:22           143.360 userdiff
04/03/2002  10:22             1.024 userdiff.LOG
04/03/2002  10:22           360.448 system.sav
04/03/2002  10:22           536.576 software.sav
04/03/2002  10:22            81.920 default.sav
04/03/2002  10:22             1.024 system.LOG
04/03/2002  14:52            65.536 DnsEvent.Evt
12/03/2002  13:37       <DIR>       .
12/03/2002  13:37       <DIR>       ..
12/03/2002  13:41            65.536 File Rep.evt
15/03/2002  15:23                 8 netlogon.dnb
15/03/2002  15:23             1.887 netlogon.dns
15/03/2002  15:33            65.536 NTDS.Evt
08/07/2002  11:09           151.552 default
08/07/2002  11:09             1.024 default.LOG
24/08/2002  19:10           524.288 AppEvent.Evt
```

Figura 4.11: Comando dir para coleta de MAC Times.

```
D:\>DIR C:\WINNT\Tasks
 Volume in drive C is 2000
 Volume Serial Number is 40D3-2B28

 Directory of C:\WINNT\Tasks

26/08/2002  21:28               184 nc -e cmd.exe 10.0.0.5 -p 1234.job
               1 File(s)            184 bytes
               0 Dir(s)     890.007.552 bytes free
```

Figura 4.12: Listagem de tarefas agendadas.

Ao contrário da *live analysis* onde as informações são coletadas, na sua maioria, em um nível mais alto de abstração, no qual os dados estão organizados em estruturas bem definidas como conexões de rede e processos, na análise *postmortem* muitas vezes é necessário lidar com o fato de que tal tipo de organização é obtida através da atuação de várias camadas de software (drivers, kernel e aplicações) que interpretam os dados originalmente armazenados na memória ou disco em forma bits. Note que muitas vezes não é interessante que haja tantas interpretações antes que a informação chegue ao examinador, evitando assim que dados potencialmente importantes sejam descartados. Um exemplo é a exclusão convencional de um arquivo, na qual o sistema operacional não destrói as informações no disco, mas apenas libera os blocos de dados

que ele ocupava para que sejam reutilizados. Logo os dados continuam presentes e são apenas "escondidos" pela camada do sistema operacional.

A análise *postmortem*, dependendo da necessidade e gravidade do incidente, deve subverter este sistema de camadas algumas vezes e coletar dados de maneira mais autônoma. Para o momento pode-se dar dois exemplos de análises *postmortem* utilizando as camadas habituais de software, nos quais é apenas prevenido o possível controle do atacante sobre o que observamos na máquina vítima:

- Registro: a análise do registro de uma máquina pode ser feita de maneira offline através dos arquivos exportados com o regedit ou regedt32, como visto na Seção 4.2.5, e podem ser lidos em qualquer editor de texto. Contudo, o examinador pode optar por proceder sua análise através de um dos dois programas supra-citados para uma melhor visualização e devido às facilidades de manipulação por eles apresentadas. Neste caso, os arquivos que foram exportados na máquina vítima podem ser importados na estação forense. Entretanto, pode haver a mistura de dados originários da máquina vítima com dados da estação forense. Desta forma, é importante ressaltar a necessidade de se efetuar *backup* do registro da estação, uma vez que seus valores serão sobrescritos. Procedimentos para a criação e recuperação de *backups* do registro do Windows 2000 podem ser obtidos em [32];

- Logs: Os logs do sistema são armazenados usualmente nos arquivos secevent.evt, appevent.evt e sysevent.evt dentro do diretório `\%systemroot%\system32\Config`, podendo ser copiados em uma segunda máquina para uma análise *offline*. A obtenção dos arquivos supracitados pode ser feita através da leitura das imagem dos discos em uma máquina Linux com suporte a montagem do sistema de arquivos NTFS, por exemplo. Uma vez obtidos os respectivos arquivos, eles podem ser importados pelo EventViewer da estação forense.

4.4 Resumo

Neste capítulo foram citadas algumas ferramentas e técnicas que podem, respectivamente, fazer parte do Kit e dos procedimentos de resposta de uma organização. A intenção não foi a produção de um manual para a composição destes dois personagens. Tratou-se apenas de uma referência para que o leitor possa compreender a abrangência e a complexidade envolvida na preparação dos programas descritos no Capítulo 2 e entender a necessidade de uma preparação técnica para

tais eventos. Na Tabela 4.3 temos agrupadas as ferramentas citadas neste capítulo e suas respectivas URLs.

Ferramenta	URL
auditpol	NTRK
dumpel	NTRK
ethereal	http://www.ethereal.com/distribution/win32/ethereal-setup-0.9.6.exe
fport	http://www.foundstone.com/knowledge/zips/FPortNG.zip
John the Ripper	http://www.openwall.com/john/john-16w.zip
l0phtcrack	http://www.atstake.com/research/lc/application/lc4setup.exe
listdll	http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml
ntlast	http://www.foundstone.com/knowledge/termsofuse.html?file-name=ntlast30.zip
psexec	http://www.sysinternals.com/files/psexec.zip
pskill	http://www.sysinternals.com/files/pskill.zip
pslist	http://www.sysinternals.com/files/pslist.zip
psloggedon	http://www.sysinternals.com/files/PsLoggedOn.zip
pstat	NTRK
pulist	NTRK
pwdump	http://www.polivec.com/Downloads/pwdump3v2.zip
rasautou	NTRK
rasuser	NTRK
reg	NTRK
regdump	NTRK
rmtshare	NTRK
windump	http://windump.polito.it/install/bin/alpha/WinDump.exe
winpcap	http://winpcap.polito.it/install/bin/WinPcap_3_0_a.exe

Tabela 4.3: Bookmark de ferramentas

Note que a abordagem feita por este capítulo pode sugerir a idéia de que uma análise forense ou uma resposta a incidentes é uma tarefa simples onde seria necessária apenas a execução de alguns poucos programas para solucionar o problema, quando na verdade a maioria das investigações são frustrantes, com dados desencontrados e inconclusivos.

Capítulo 5

Análise do Sistema de Arquivos NTFS

Dando seqüência ao estudo de metodologias de análise forense para ambiente Windows, será abordada neste capítulo a análise do sistema de arquivos NTFS. A preocupação com a análise do sistema de arquivos se justifica pela dificuldade de se atacar uma máquina sem que este personagem seja alterado, transformando-o em uma importante fonte de evidências.

A Seção 5.1 é constituída pelo artigo "*Metodologias de Análise Forense para Ambientes Baseados em NTFS*", publicado nos anais do III Simpósio de Segurança em Informática (SSI'2001) [45]. Nas seções subseqüentes 5.2, 5.3 e 5.4 complementa-se a abordagem feita no artigo, procurando exibir ferramentas e técnicas ainda não discutidas no momento de sua publicação.

O objetivo deste capítulo é fornecer ao leitor uma visão geral acerca da estrutura do sistema de arquivos nativo do Windows 2000, além de discutir técnicas e ferramentas que poderiam ser utilizadas durante sua análise. Através do exame minucioso do NTFS em baixo nível é possível visualizar dados e estruturas que são geralmente ocultados pela interface entre o SO e o usuário, o que pode, conseqüentemente, levantar indícios que criem, comprovem ou descartem teorias sobre o incidente.

5.1 Artigo

METODOLOGIAS DE ANÁLISE FORENSE PARA AMBIENTES BASEADOS EM NTFS

Flávio de Souza Oliveira
Instituto de Computação
Universidade Estadual de Campinas
13083-970 Campinas - SP
flavio.oliveira@ic.unicamp.br

Célio Cardoso Guimarães
Instituto de Computação
Universidade Estadual de Campinas
13083-970 Campinas - SP
celio@ic.unicamp.br

Paulo Lício de Geus
Instituto de Computação
Universidade Estadual de Campinas
13083-970 Campinas - SP
paulo@ic.unicamp.br

RESUMO

A definição de técnicas para a identificação e coleta de evidências digitais é essencial para uma possível ação judicial contra o autor de um ataque bem sucedido. Este fato, faz com que a criação de metodologias para identificar evidências no sistema de arquivos se torne um dos pontos mais importantes de uma análise forense computacional.

ABSTRACT

The techniques' definition for digital evidences identification and acquisition is essential for a possible prosecution against the author of a successful attack. This fact implies that the methodology's creation for evidences identification in a file system becomes one of the most importants point in forensics analysis.

5.1.1 Introdução

Atualmente a grande maioria das instituições conectadas à Internet conta com algum aparato de segurança, com o intuito de evitar que sua rede se torne alvo fácil para toda sorte de atacante e bisbilhoteiro que prolifera pela Rede. Entretanto, aparatos como *firewalls* e VPN's são compostos por peças de software que contêm inúmeras linhas de código, e que por sua vez, não estão imunes a erros de programação. Logo, mesmo que uma organização tome todos os cuidados para manter a segurança dos seus dados, não está totalmente livre da possibilidade de ser vítima de um ataque bem sucedido.

A definição de uma política a ser adotada no caso de um incidente de segurança é essencial para que os danos sejam minimizados. É necessário que haja metodologias para que uma vez descoberta a invasão, seja possível identificar, coletar e manipular evidências sem distorcê-las, mantendo-se assim a possibilidade de futuramente adotar-se medidas legais contra o invasor.

Nesse artigo serão discutidas algumas metodologias e problemas na aquisição de evidências relacionadas ao sistema de arquivos de plataformas Windows 2K/NT (NTFS), sistema amplamente utilizado, porém carente de estudos com tal abordagem.

O objetivo é expor algumas técnicas e aspectos que devem ser considerados durante uma análise forense do sistema de arquivos NTFS, a fim de evitar a dependência de elementos privados de software e metodologias que não sejam de domínio público.

5.1.1.1 Ciência Forense

Ao abordar o termo forense, se é automaticamente remetido ao meio policial, onde, na tentativa de solucionar um mistério, policiais e peritos devem analisar minuciosamente todo tipo de objetos, sinais e marcas que estejam presentes na cena do crime.

A análise forense inicia imediatamente após a chegada dos policiais ao local, começando pelo isolamento eficiente do perímetro, evitando assim, a exposição excessiva e possível contaminação das evidências. Passa-se então para a fase de identificação e coleta de todo tipo de dado e material que possa ter alguma relevância na resolução do caso em questão. Apenas após a realização dessas duas primeiras etapas inicia-se uma análise laboratorial das possíveis evidências, tais como: análise balística e de DNA. Tal fato contraria o que muitas pessoas pensam ao considerar apenas a análise laboratorial como análise forense [26].

O sucesso da análise está então ligado diretamente ao sucesso de todas as três etapas que constituem o processo. Caso haja contaminação ou falhas na aplicação de metodologias para aquisição ou manipulação de evidências, tem-se por sua vez, uma grande possibilidade de não se conseguir resultados precisos, ou mesmo resultado algum durante uma análise laboratorial, isto avaliando apenas aspectos técnicos, uma vez que falhas desse tipo invalidam completamente uma evidência em um tribunal.

As técnicas envolvidas na análise laboratorial de uma evidência são divididas em etapas que dependem diretamente do tipo de material que se está analisando, o que pode variar desde um cadáver a uma minúscula mancha de tinta. Tome-se por exemplo a análise feita no DNA reco-

lhido de uma amostra de sangue na cena de um crime. É possível aplicar exatamente o mesmo protocolo a toda amostra de DNA recebida: eliminam-se as impurezas e o reduz à sua forma elementar [41]. Todos estes procedimentos devem ser padronizados, gerar resultados reproduzíveis e serem aceitos pela comunidade científica internacional.

Uma vez descrita a origem do termo forense, bem como exemplificada a utilização e as etapas do processo de análise em um âmbito de investigação geral, na Seção 5.1.2, é feita uma introdução à aplicação desta ciência no meio computacional.

5.1.2 Forense Computacional

Com o advento do computador e o surgimento dos primeiros casos envolvendo o meio computacional, tornou-se necessária a criação de uma nova disciplina forense, que deveria preocupar-se em atuar nesse novo nicho, criando metodologias e acumulando conhecimentos para a aquisição, manipulação e análise de evidências digitais.

A resolução de um mistério computacional pode ser uma tarefa árdua e difícil. É necessário que se examine o sistema minuciosamente, assim como um detetive examina a cena de um crime [19]. Para isso a pessoa que está realizando a análise deve conhecer profundamente o sistema operacional em que está trabalhando, podendo então identificar e entender as relações de causa e efeito de todas as ações tomadas durante a análise.

Tal nível de conhecimento é essencial, devido à enorme volatilidade de certos tipos de evidências. Tome-se por exemplo os tempos de acesso, criação e modificação de um arquivo, conhecidos como MACTimes. A simples seleção de um arquivo no *explorer* do Windows altera o tempo do último acesso, anulando assim um dos mais poderosos mecanismos de se reconstituir o que ocorreu no sistema em um passado recente. Voltar-se-á a falar de MACTimes nas seções que se seguem, bem como dos problemas de sua utilização em ambientes Windows.

Uma vez entendidas as relações de causa e efeito dentro do sistema, existe ainda a necessidade de uma série de habilidades para que um perito possa conduzir uma análise forense de maneira eficaz. Segundo Venama e Farmer [19], felizmente muitas dessas habilidades são características aos programadores, tais como: raciocínio lógico, possuir uma mente aberta e o entendimento das relações de causa e efeito. Tais habilidades são largamente utilizadas durante a busca de um erro em um programa. Contudo a depuração de um programa ainda está distante do desafio representado por uma análise forense, já que ao se depurar um programa está

lutando-se contra si mesmo, enquanto em uma análise forense enfrenta-se outro programador que não tem o interesse de ser descoberto [19].

5.1.2.1 Metodologias no Tratamento de Evidências Digitais

No estágio atual das pesquisas no campo da forense computacional ainda existe muita carência de metodologias para o manuseio desse tipo de evidência. Tal carência pode ser explicada pelo fato de existirem inúmeras mídias e sistemas operacionais, além de diversas mudanças de versão. Todos esses fatores tornam difícil a definição de padrões e metodologias, pelo menos da forma como acontece com as outras disciplinas forenses [44].

Atualmente já existem padrões definidos sendo aplicados de forma experimental [57]. Eles foram desenvolvidos pelo SWGDE (*Scientific Working Group on Digital Evidence*), que é o representante norte-americano na *International Organization on Computer Evidence* (IOCE). Tais padrões foram apresentados durante a *International Hi-Tech Crime and Forensics Conference* (IHCFC), realizada em Londres, de 4 a 7 de outubro de 1999.

Os padrões desenvolvidos pelo SWGDE seguem um único princípio: o de que todas as organizações que lidam com a investigação forense devem manter um alto nível de qualidade a fim de assegurar a confiabilidade e a precisão das evidências. Esse nível de qualidade pode ser atingido através da elaboração de SOPs (*Standard Operating Procedures*), que devem conter os procedimentos para todo tipo de análise conhecida e prever a utilização de técnicas, equipamentos e materiais largamente aceitos na comunidade científica internacional [41].

Após a compreensão da complexidade envolvida no manuseio de evidencias digitais, pode-se então identificar alguns métodos para manipulação de evidências relacionadas ao sistema de arquivos, como proposto na Seção 5.1.1

5.1.2.2 Manipulação do Sistemas de Arquivos

O desenvolvimento de documentos como os SOPs dependem de técnicas específicas para cada tipo de situação e sistema operacional (SO). No entanto, alguns procedimentos relacionados à manipulação de sistemas de arquivos são gerais e não dependem do SO que está sendo analisado. Tais procedimentos já são adotados pela comunidade forense internacional. Dentre eles cita-se:

- *Análise sobre cópias*: Ao se iniciar uma análise deve-se considerar veementemente a atuação a partir de cópias dos dados originais, mas não cópias comuns (`copy`). O ideal é que as cópias sejam feitas bit a bit (imagem). Dessa forma, copia-se todos os blocos de dados, inclusive os de arquivos apagados, sem a deturpação dos tempos de acesso. Cópias desse tipo podem ser feitas através de ferramentas semelhantes ao comando `dd` do UNIX. A Seção 5.1.6, aborda algumas ferramentas que podem ser úteis durante a análise de sistemas baseados em NT.

- *Assinaturas digitais*: Para que não paire dúvidas sobre a confiabilidade das imagens e conseqüentemente das evidências obtidas, deve-se garantir que os dados analisados são exatamente iguais aos dados originais recolhidos da "cena do crime". Uma maneira de assegurar essa confiabilidade é através de assinaturas digitais, geralmente MD5.

- *Sem permissão de escrita e execução*: Um fato importante a se atentar antes de começar uma análise, é o de se examinar a imagem sem permissão de escrita e até mesmo de execução, a fim de evitar a alteração ou a execução involuntaria de algum arquivo. Muitos sistemas também permitem desabilitar a atualização dos tempos de acesso, que pode eventualmente ser interessante.

- *MACTimes*: São potencialmente uma das mais poderosas formas de se reconstituir o que ocorreu em um sistema de arquivos no passado [18]. Esse termo é usado para referenciar os três atributos de tempo presentes em todos os arquivos e diretórios da maioria dos SO's (mtime, atime e ctime). No caso do 2000 esses atributos são chamados de *LastWriteTime, LastAccessTime e CreationTime*. A análise desses atributos envolve a utilização de ferramentas especiais que utilizam chamadas de sistema diretamente ou contornem de alguma forma os métodos convencionais de acesso a arquivos, com o intuito de evitar a perda de informações. Um problema apresentado por este tipo de evidência é a sua volatilidade e sua pouca utilidade em ambientes de grande atividade [18].

- *Arquivos excluídos*: Um mecanismo eficiente de recuperação de arquivos excluídos também pode ser decisivo durante uma análise forense, uma vez que na tentativa de esconder seus rastros, os invasores podem apagar arquivos que possam denunciar sua presença. O problema é que existem inúmeras ferramentas para se efetuar uma exclusão de forma segura: são as chamadas ferramentas de *wipping*.

5.1.3 Forense em ambiente NT

A análise forense de sistemas baseados em Windows NT apresenta-se como um grande desafio para forense computacional, pois trata-se de um sistema operacional fechado, de documentação escassa e controversa, mas largamente utilizado.

Atualmente sabe-se que o ideal é que a análise seja feita em um ambiente controlado, isto é, isolado e com ferramentas confiáveis, de preferência utilizando um outro sistema operacional para que se evitem vícios comuns a ambientes Windows, tal como o principio de tornar tudo o mais fácil possível para o usuário. Todavia, o NTFS contém muitas peculiaridades não suportada por *drivers* de outros sistemas, entre eles o Linux.

Um problema a ser resolvido, é a não existência, no ambiente NT, de estudos sobre as ações tomadas por seus atacantes, como acontece no UNIX. Um exemplo desse tipo de documentação é o projeto honeynet[1], que conta com diversas descrições de incidentes, bem como a descrição das ações tomadas durante suas investigações.

Sistemas UNIX também contam com diversas ferramentas de código aberto, o que facilita o entendimento das metodologias adotadas. Dentre elas podemos citar o *The Coroners ToolKit* (TCT), conjunto de ferramentas reunidas por Dan Farmer e Wietse Venema que já está se tornando um padrão de fato na análise forense desse sistema.

5.1.4 Estrutura do NTFS

O NTFS (*New Technology File System*) é o sistema de arquivos nativo do Windows NT/2K, embora mantenham suporte ao sistema FAT originário do DOS. Ele foi desenvolvido com o objetivo de suprir as necessidades do mercado coorporativo, tais como: maior capacidade de endereçamento, suporte a critérios de segurança aplicáveis a cada arquivo individualmente, cifragem de dados entre outros.

A formatação de uma partição NTFS resulta na criação da *Master File Table* (MFT) e de diversos arquivos de sistema [38]. A MFT contém informações sobre todos os arquivos e diretórios de uma partição NTFS. A Figura 5.1 ilustra o posicionamento padrão da MFT.

Além da MFT, o processo de formatação ainda cria um conjunto de arquivos que contém meta informações usadas para implementar a estrutura do sistema de arquivos. Tais arquivos são

1. http://project.honeynet.org/

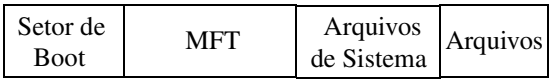


Figura 5.1: Estrutura de um Volume NTFS

mapeados nos primeiros registros da MFT, inclusive a própria. A Tabela 5.1 mostra tais arquivos e seu mapeamento junto à MFT.

File Name	MFT Record	Descrição
$MFT	0	Master File Table
$MftMirr	1	Cópia dos 16 primeiros registros da MFT
$LogFile	2	Arquivo de log das trasações efetuadas no disco
$Volume	3	Número de série do volume, data de criação e o *dirty flag*
$AttrDef	4	Definição dos atributos
$.	5	Diretório raiz do volume
$Bitmap	6	Representação do disco indicando que clusters estão sendo utilizados
$Boot	7	Setor de *boot* do volume
$BadClus	8	*Clusters* defeituosos
$Secure	9	Contém *security descriptors* únicos para todos os arquivos do volume
$Upcase	10	Mapeia caracteres minúsculos em seus correspondentes maiúsculos
$Extend	11	Usado por várias extensões opcionais, como quotas e reparse points
	12-15	Reservados para uso futuro.

Tabela 5.1: Arquivos de Meta informação

Até o NT 4.0 tais arquivos podiam ser vistos através do comando `dir`, como na Figura 5.2.

```
C:\> dir /ah <nome do arquivo>
```

Figura 5.2: Visulização de Arquivos de Meta informação no NT 4.0.

5.1.4.1 Master File Table

Como dito anteriormente a MFT contém informações sobre todos os arquivos e diretórios do volume. Cada registro é composto por um pequeno cabeçalho que contém informações básicas descrevendo o próprio registro, como listado abaixo:

- Números de seqüência, usados para verificação de integridade;
- Ponteiro para o primeiro atributo do registro;
- Ponteiro para o primeiro byte livre no registro;
- Número do registro em relação ao registro base da MFT, caso não seja o primeiro.

O cabeçalho inicial é seguido por um ou mais atributos que descrevem as características do arquivo. A Figura 5.3 ilustra a estrutura de registro padrão.

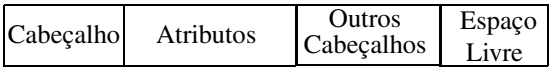


Figura 5.3: Registro da MFT

Cada atributo é dividido em dois componentes: um cabeçalho que guarda o tipo do atributo, nome, *flags* e a localização da parte de dados do registro, e uma parte de dados onde é armazenada a informação do registro. Existe uma série de possibilidades para o armazenamento de dados nos registros da MFT, mas tais detalhes estão fora do escopo deste artigo. Na Figura 5.4 tem-se um exemplo de registro, onde pode-se observar seus primeiros atributos, dentre os quais destaca-se o atributo *Standard Info*, que armazena os MACTimes. Na Seção 5.1.4.2 tem-se uma discussão da análise desse tipo de evidência.

5.1.4.2 MACTimes

Uma vez compreendida a estrutura básica de um volume NTFS, pode-se então partir para uma análise mais detalhada sobre os MACTimes do ambiente Windows.

Um primeiro problema que aflige a grande maioria dos SO's é a falta de um registro histórico, uma vez que os tempos registrados nos arquivos dizem respeito apenas à última modificação o que torna impossível a obtenção dos acessos anteriores utilizando-se apenas o sistema de arqui-

vos [18]. Uma solução para esse problema poderia ser a habilitação de um log que registra os acessos aos arquivos críticos do sistema, o que além de fornecer um histórico dos acessos ainda forneceria outros dados como o usuário que fez o acesso. Tal solução todavia, pode ter efeitos colaterais: como o possível excesso de logs, além do fato de que seria necessário um estudo para se descobrir quais arquivos monitorar.

<table>
<tr><td rowspan="2">Header</td><td rowspan="2">Name</td><td rowspan="2">Arquivo.txt</td><td rowspan="2">Standard Info</td><td rowspan="2">January 3, 2000 8:30 pm
January 3, 2000 8:46 pm
December 9, 1999 5:03 pm</td><td>Attribute List</td><td rowspan="2">Data (non-resident)</td></tr>
<tr><td>Data</td></tr>
</table>

Figura 5.4: Exemplo de Registro da MFT

Outro problema é que para efeitos de desempenho, o *LastAccessTime* tem resolução de uma hora, logo, os acessos efetuados em um intervalo menor de tempo, aparentemente não são registrados. No entanto, existem relatos de que é possível se verificar tempos de acesso com resolução de segundos. Entretanto, tal possibilidade foi anunciada por membros de uma empresa da área que não divulgaram os detalhes da operação.

A análise de MACTimes no Windows ainda possui uma série de anomalias, tais como:

- Quando se copia um arquivo para um outro de nome diferente, a data da última modificação continua igual à do arquivo original, enquanto as datas do último acesso e criação se comportam normalmente, dando a impressão que o arquivo foi modificado antes de ser criado;
- Recentemente, na lista de discussão sobre forense da Security Focus[1] foi descrito o seguinte problema: ao se copiar um arquivo para um outro com o mesmo nome de um recém excluído, a data de criação do novo arquivo assume a do arquivo antigo, excluído anteriormente. Neste caso, provavelmente trata-se de um erro de programação e não de projeto. Entretanto, tal anomalia poderia causar deturpações em um processo de análise.

5.1.5 Alternate Streams

As *alternate streams* são sem sombra de dúvida um dos pontos cruciais da análise de um sistema NTFS. Elas foram tratadas como falhas de segurança primeiramente pelos colunistas da InfoWorld Security, Stuart McClure e Joel Scambray em julho de 98. Em linhas gerais, trata-se de um

1. http://www.securityfocus.com/forums/forensics/intro.html

mecanismo para embutir um arquivo dentro de outro, sem que seu conteúdo ou tamanho seja alterado. Tecnicamente falando, podemos dizer que todo arquivo NTFS possui um outro arquivo sem nome embutido, chamado *default stream* ou *unnamed stream*, onde os dados convencionais, como texto e programas, são armazenados. Todavia, existe a possibilidade de criar-se arquivos embutidos com nomes diferentes, são os chamados *alternate streams* [24][53].

No início dos anos 90 a Microsoft introduziu essa funcionalidade com o intuito de tornar o NT um servidor de arquivos para computadores que utilizassem o Mac OS. O objetivo era simular os *resource forks* do HFS (*Hierarchical File System*) usado para armazenar dados como ícones e outros tipos de metainformação.

O Windows utiliza arquivos *alternate streams* para armazenar informações fornecidas na *summary tab* do explorer, como mostrado na Figura 5.5. Tais informações são armazenadas em uma *stream* chamada *?SummaryInformation*, onde o ponto de interrogação indica um caracter especial que não pode ser impresso.

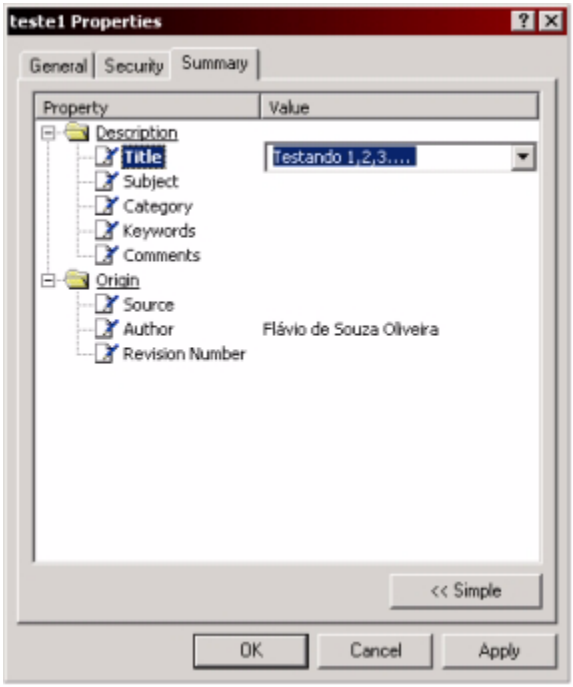


Figura 5.5: Summary Tab

5.1.5.1 Acesso ao Conteúdo

O acesso aos arquivos embutidos pode ser feito através da notação da Figura 5.6. Entretanto, poucos comandos podem utilizar tal notação, entre eles podemos destacar o `more` cuja utilização é demonstrada na Figura 5.7.

```
-> <nome_arquivo>:<nome_stream>

-> leiame.txt:cmd
```

Figura 5.6: Notação.

```
C:\> echo Testando > arq.txt:otr

C:\> more < arq.txt:otr
```

Figura 5.7: Acesso.

Com a execução do comando `echo` cria-se um arquivo embutido chamado *otr* e com o `more` podemos verificar seu conteúdo. Note que o acesso é feito através de redirecionamento, uma maneira de contornar a falta de suporte dos comandos.

Uma vez demonstrada o acesso a um arquivo embutido, uma pergunta é quase imediata: "É possível embutir executáveis?". A resposta a essa pergunta é sim, como pode ser visto na Figura 5.8.

```
C:\>type nc.exe > otk.exe:a.exe
```

Figura 5.8: Embutindo Executáveis

Um programa localizado em uma *alternate stream* pode ser executado de 5 formas diferentes no Windows 2000 [24]:

- Através da opção *run* do menu *start* do Windows utilizando-se a notação da Figura 5.9.

```
-> file:\\<dir>\<arq>:<stream>

-> file:\\c:\otk.exe:a.exe
```

Figura 5.9: Execução Através do Run

- Com a utilização de atalhos, inclusive na pasta *startup*.
- Inclusão de uma chave na pasta do registro que gerencia os programas executados automaticamente durante o *logon*.
- Para *vb scripts* existe uma possibilidade extra, através do comando `wscript` (Figura 5.10).

```
-> wscript <arquivo>:<stream>

C:\> wscript otk.exe:s.vbs
```

Figura 5.10: Execução Através do Wscript

- E por fim, através de um programa criado com esse objetivo. (Figura 5.11)

5.1.5.2 Perigos e Possibilidades

Em agosto de 2000, dois "*hackers*" tchecos criaram o W2K.Stream [24], um vírus que utilizava os arquivos embutidos do Windows 2000. Apesar de não causar maiores danos à máquina infectada o W2K chamou novamente a atenção para o problema.

No meio forense o perigo está na possibilidade de se ocultar um programa malicioso como o `netcat` (veja Seção 5.1.6). Dessa forma, tem-se uma *backdoor* que não poderia ser localizada no disco através das ferramentas convencionais.

Apesar de tratar-se de uma funcionalidade bem conhecida, arquivos embutidos não criam alterações nas assinatura gerada pela maioria dos elementos de software para assinatura digital. Sendo assim arquivos com *alternate streams* só podem ser descobertos através da utilização de programas específicos, como o streams do Mark Russinovich citado na Seção 5.1.6.

5.1.6 Ferramentas Úteis

Nesta seção estão relacionadas algumas ferramentas que podem ser úteis ao se efetuar uma análise. Todas são grátis, mas nem todas de código aberto. Seguem-se:

- CygWin Tools: Versão para Windows de diversas ferramentas GNU, inclusive o `dd`. Isto é possível, graças às bibliotecas `cygwin` que simulam as chamadas de sistema e geram o ambiente que tais ferramentas necessitam. Endereço:
  http://sources.redhat.com/cygwin/;

```
#! c:\perl\bin\perl.exe

use strict;
use Win32API::File 0.02 qw( :ALL );

my $server = shift || Win32::NodeName;
my $path;

($server eq Win32::NodeName)?($path = c:\\winnt\\):
($path = \\\\$server\\c\$\\winnt\\);

my $astream = Win32API::File::CreateFile($path.lanmannt.bmp:test.txt,
GENERIC_READ|GENERIC_WRITE, FILE_SHARE_WRITE, [],
CREATE_NEW,[],[]);

my $str = This is a test of alternate data streams;

if ($astream) {
Win32API::File::WriteFile($astream, $str, length($str), [], []);
Win32API::File::CloseHandle($astream);
print Alternate file stream created.\n;
}

else {
print Alternate file stream creation failed: $^E\n;
}
```

Figura 5.11: Exemplo de Código para Criar Alternate Streams [9]

- Foundstone Forensic ToolKit: Conjunto de ferramentas de código aberto para análise forense de arquivos em um volume NTFS. Endereço: http://www.foundstone.com/rdlabs/termsofuse.php?filename=ForensicToolkit20.zip;

- GNU Unix Utils for Win32: Versão para Windows de diversas ferramentas GNU, dentre elas o `dd` e o `md5sum`. A vantagem dessas ferramentas é o fato de necessitarem apenas da biblioteca `msvcrt.dll`. Ao contrário das CygWin Tools, que necessitam de toda uma camada extra de software para funcionarem. Endereço: ftp://ftp.uni-koeln.de/pc/win32/misc/unxutils.zip;

- MS-DiskEdit: Ferramenta acidentalmente inclusa no SP4 do Windows NT 4.0. Aparentemente era uma ferramenta de depuração interna, usada pelos desenvolvedores do NTFS. O importante é que ela pode acessar o disco em baixo nível (hexadecimal); e entende a estrutura da MFT, interpretando e exibindo o conteúdo dos registros de forma organizada e legível. O `MS-DiskEdit` pode ser obtido através do seguinte endereço: http://www.informatik.fh-hamburg.de/pub/nt-service/sp4en-ex/i386/diskedit.exe; [52]

- NetCat: Versão para NT do `netcat` original do UNIX. Muito usado como *backdoor* pelos atacantes. Endereço: http://www.l0pht.com/users/10pht/nc11nt.zip;
- Perl Scripts: O endereço mencionado abaixo possui uma série de *scripts* Perl que podem ser úteis durante uma análise do sistema de arquivos, além de outras funcionalidades. Endereço: http://patriot.net/~carvdawg/perl.html
- Streams: Programa que detecta a existência de *alternate streams* em um arquivo. Endereço: http://www.sysinternals.com/files/streams.zip;
- Strings: Recupera todas as cadeias de caracteres contidas em um arquivo executável. Muito útil durante a identificação da função de um dado software. Endereço: http://www.sysinternals.com/files/strings.zip;

5.1.7 Conclusões

A análise forense de um sistema proprietário como o Windows NT/2000 torna difícil a definição de metodologias totalmente confiáveis. Uma vez que não se tem a exata noção das conseqüências de ações tomadas durante a análise. Torna-se então indispensável a troca de informações e experiências a fim de se conseguir um conjunto de metodologias amplamente aceitas, de forma a não se correr o risco da dependência de metodologias fechadas e elementos privados de software.

5.2 Slack Space

Os *slack spaces* podem representar importantes fontes de informação durante uma investigação mais detalhada, pois contém trechos aleatoriamente selecionados da memória, que por sua vez, podem conter informações a respeito de *logon* de usuários, senhas utilizadas para a cifragem de arquivos, além de fragmentos de comunicações textuais via Internet tal como e-mail, sessões de *chat* e *newsgroups*.

Existem dois tipos de *slack spaces*:

- ***File Slack***: Os sistemas operacionais da Microsoft armazenam seus arquivos em disco utilizando blocos de dados de tamanho fixo chamados *clusters* (Figura 5.12), contudo, os arquivos em um disco podem ter os mais variados tamanhos, dependendo do seu conteúdo. Desta forma, raramente o tamanho de um arquivo é múltiplo do tamanho de um

cluster, o que impede o armazenamento ideal. Sendo assim, é comum que o último *cluster* associado a um arquivo não seja totalmente utilizado por ele, permitindo que dados excluídos deste e de antigos arquivos possam ser capturados e analisados.

- ***RAM Slack***: Além de dados de antigos arquivos do disco, o *file slack space* também pode conter conjuntos de bytes aleatoriamente selecionados da memória RAM. Isto ocorre porque o Windows normalmente efetua escritas no disco em blocos de 512 bytes, chamados setores. Como normalmente a quantidade de informação a ser gravada não pode ser dividida igualmente em blocos de 512, geralmente é necessário que o último bloco de informação tenha que ser completado de alguma forma (Figura 5.12). Neste caso, o Windows faz o complemento com os *buffers* de memória do sistema operacional.

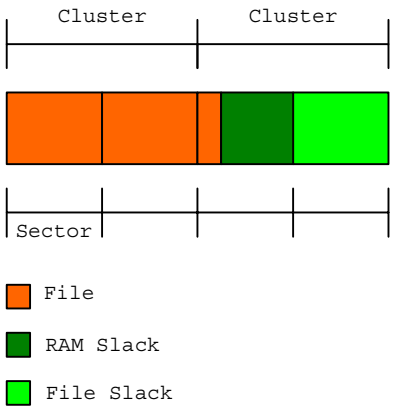


Figura 5.12: Slack Space

Para a recuperação das informações contidas no *slack space* é necessário a utilização de programas para acesso ao disco em baixo nível, como o `MS-DiskEdit`, contudo, existem ferramentas comerciais especializadas na obtenção de tais dados, o que torna o trabalho bem mais fácil, como por exemplo o `GetSlack`[1] da New Technologies Inc.

É importante ressaltar que existem também ferramentas para eliminar este tipo de informação. Em 1995 o departamento de defesa americano em conjunto com algumas outras agências desenvolveu uma série de recomendações para auxiliar as empresas a não serem vítimas de espionagem industrial. Dentre os tópicos que tratam de informação digital, os *slack spaces* não são

1. http://www.forensics-intl.com/getslack.html

citados diretamente, mas se incluem na seguinte recomendação: "*Overwriting all addressable locations with a character, its complement, then a random character and verify.*". Este documento incentivou a criação de diversos aplicativos para a exclusão segura de arquivos[1] e para limpeza dos *slack spaces*, mas pouco utilizados no dia-a-dia o que faz com que sua execução produza anomalias como as observadas na Seção 5.4. Um exemplo deste tipo de programa é o `Noise Wiper`[2]. [14]

5.3 Linux como Base de Coleta de Evidências

A utilização de uma outra plataforma para coleta de evidências pode representar uma alternativa interessante para o examinador, pois dessa forma é possível evitar vícios de visualização através da observação dos dados sob um novo "ponto de vista". Com base nesta teoria, o Linux se apresenta como uma boa opção, devido, principalmente, à existência de um projeto[3] voltado ao desenvolvimento de um *driver* para o acesso a volumes NTFS (volumes FAT já são amplamente suportados) em boa fase de desenvolvimento, permitindo montagens seguras, apenas para leitura até o momento.

Um outro fator favorável à adoção do Linux para parte das análises, é a grande disponibilidade de ferramentas de código aberto, além da possibilidade de utilização de aplicativos já consagrados no meio forense, como algumas ferramentas do TCT. Contudo, o suporte ao NTFS fornecido pelo *driver* do Linux não é completo, o que implica na desconsideração de estruturas importantes como as *Alternate Streams*.

Outra ferramenta para Linux, que agora suporta imagens[4] de volumes NTFS, é o TASK[5] (*The @stake Sleuth Kit*) versão superior a 1.50. Os aplicativos que compõem o TASK tiram todo proveito da versatilidade do Linux em trabalhar com imagens de disco e podem fazer análise em baixo nível de dados da MFT e das propriedades e conteúdo de arquivos, incluindo *alternate streams*. Em conjunto com o Autopsy Forensic Browser (Figura 5.14), é possível a visualização de dados via interface HTML o que torna o processo bem mais amigável.

---

1. http://www.sysinternals.com/ntw2k/source/sdelete.shtml
2. http://www.simtel.net/pub/dl/56403.html
3. http://sourceforge.net/projects/linux-ntfs
4. Cópias bit-a-bit produzidas com `dd`
5. http://prdownloads.sourceforge.net/sleuthkit/task-1.50.tar.gz?download

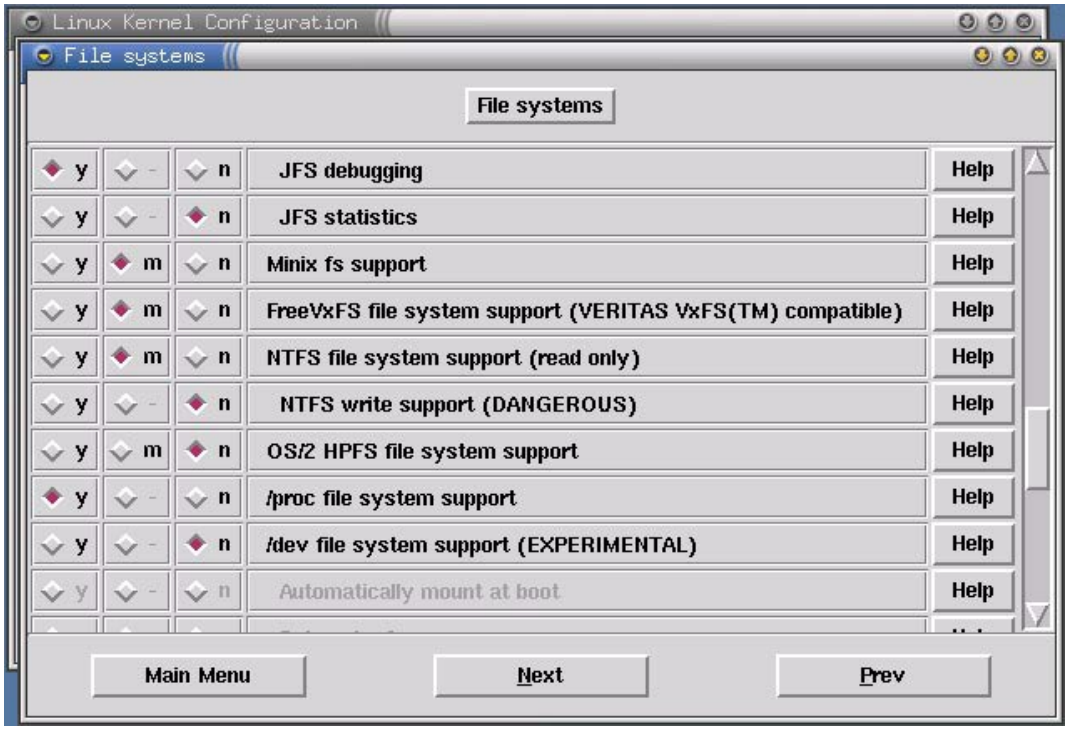


Figura 5.13: Suporte ao NTFS no kernel do Linux

Pode-se concluir então que o Linux, em conjunto com o TASK, Autopsy e o suporte fornecido pelo *driver* NTFS (Figura 5.13), pode ser uma plataforma efetiva na visualização e coleta de evidências em baixo nível, especialmente devido a versatilidade do seu mecanismo de montagem de arquivos de imagens. No entanto, quando é necessária a interpretação da sintaxe ou semântica de arquivos e informações ele pode ser de pouca utilidade.

5.4 Notas:

- O algoritmo de alocação de espaço da MFT faz com que a reutilização de suas antigas entradas seja mais rápida que a reutilização de *inodes* por parte dos sistemas Unix, o que resulta em uma perda mais rápida das informações sobre os atributos dos arquivos excluídos; [7]
- Arquivos não tem seus MACtimes alterados quando são excluídos;

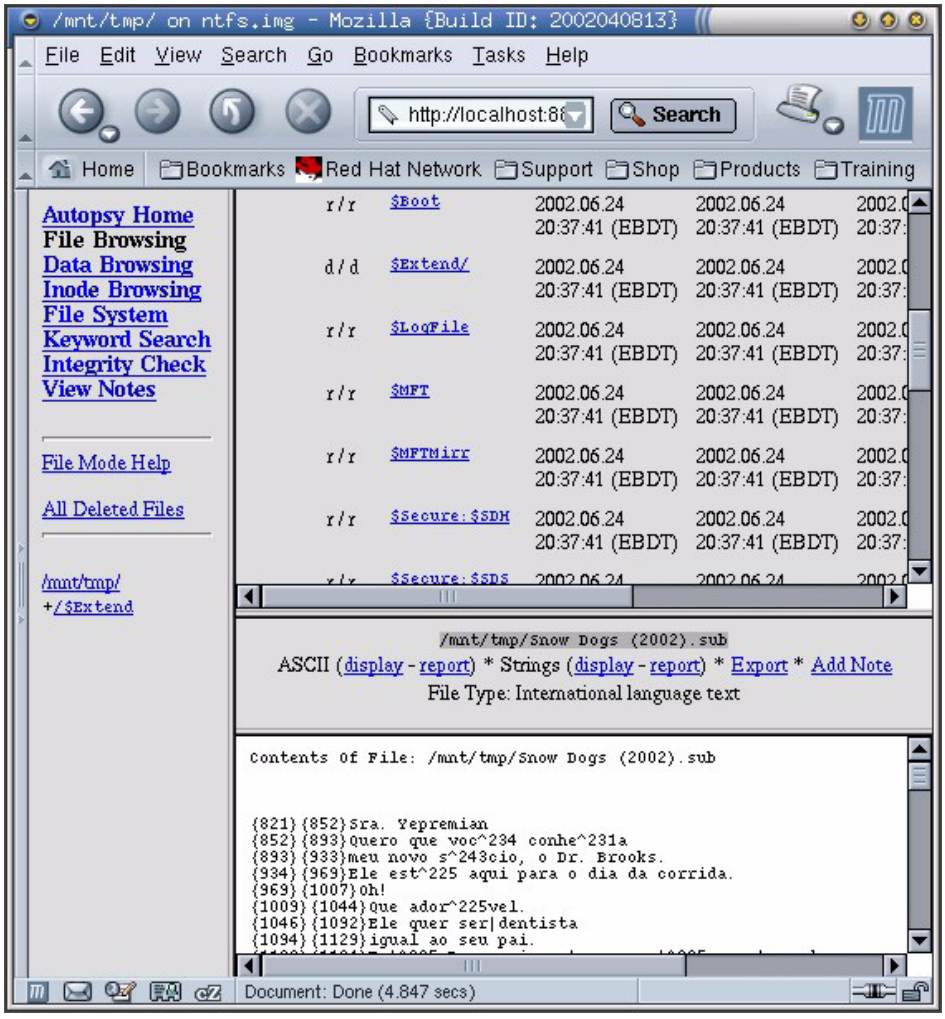


Figura 5.14: Autopsy exibindo informações de uma imagem NTFS

- A informação a respeito do nome dos arquivos que foram excluídos é frequentemente perdida durante os processos de ordenação da árvore de diretórios, diferentemente do que ocorre no FAT ou nos Sistemas Unix em geral;
- Entradas vazias no meio da MFT podem indicar a utilização de ferramentas de exclusão segura ou de *wipping*;
- Se as informações de tempo presentes no atributo *Standard Info* forem diferentes das que estiverem presentes no atributo *File Name*, ferramentas de exclusão segura ou de *wipping* podem ter sido utilizadas;

- É comum encontrar arquivos com tempos de modificação anteriores aos de criação;

5.5 Resumo

Neste capítulo foram citadas algumas ferramentas e técnicas para a obtenção de evidências durante a análise do sistema de arquivos NTFS, assim como o alerta a respeito de estruturas que podem conter dados de interesse em uma investigação. Na Tabela 5.2 temos um resumo das ferramentas citadas neste capítulo com suas respectivas URLs.

Ferramenta	URL
Strings	http://www.sysinternals.com/files/strings.zip
SDelete	http://www.sysinternals.com/files/sdelete.zip
Streams	http://www.sysinternals.com/files/streams.zip
NetCat	http://www.l0pht.com/users/10pht/nc11nt.zip
DiskEdit	http://www.informatik.fh-hamburg.de/pub/nt-service/sp4en-ex/i386/diskedit.exe
GNU Unix Utils for Win32	ftp://ftp.uni-koeln.de/pc/win32/misc/unxutils.zip
Foudstone Forensic ToolKit	http://www.foundstone.com/rdlabs/termsofuse.php?filename=ForensicToolkit20.zip
CygWin Tools	http://sources.redhat.com/cygwin/
Noise Wiper	http://www.simtel.net/pub/dl/56403.html
GetSlack	http://www.forensics-intl.com/getslack.html
The @stake Sleuth Kit (TASK)	http://prdownloads.sourceforge.net/sleuthkit/task-1.50.tar.gz?download
Autopsy Forensic Browser	http://prdownloads.sourceforge.net/autopsy/autopsy-1.60.tar.gz?download
The Coroner's ToolKit (TCT)	http://www.porcupine.org/forensics/tct.html

Tabela 5.2: Bookmark de Ferramentas

Capítulo 6

Preparação de Redes Windows para Futuras Análises Forense

Neste capítulo é apresentada a necessidade de manter as máquinas de uma rede Windows configuradas, levando em consideração o fato de que uma futura análise forense pode ser necessária. Entretanto, para isso esbarra-se em problemas como a identificação de que pontos e de quais medidas poderiam facilitar uma posterior análise. Além disso, é importante encontrar meios para que tais medidas e configurações sejam feitas de maneira automatizada e centralizada a fim de reduzir a complexidade e o custo da implantação de programas de resposta a incidente.

Visando solucionar estes problemas, desenvolvemos o PFSAF, apresentado na Seção 6.1 e que, por sua vez, é composta pelo artigo "*Pre-Forensic Setup Automation for Windows 2000*" aceito no *IASTED International Conference on Communications and Computer Networks* (CCN 2002) realizada no Massachusetts Institute of Technology (MIT) em Cambridge nos EUA, de 4 a 6 de novembro. [48]

6.1 Artigo

PRE-FORENSIC SETUP AUTOMATION FOR WINDOWS 2000

Flávio de Souza Oliveira
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
flavio.oliveira@ic.unicamp.br

Célio Cardoso Guimarães
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
celio@ic.unicamp.br

Paulo Lício de Geus
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
paulo@ic.unicamp.br

ABSTRACT
This work presents a framework for automation of administrative tasks and deployment of protection mechanisms to facilitate a future forensic analysis. The main goal is to disclose and supply measures for a fast configuration of Microsoft Windows 2000 networks, when deploying incident response procedures.

KEY WORDS
Security, Forensics, Incident Response, File Integrity, Windows 2000, Automation

6.1.1 Introduction

Establishing procedures for incident response by companies interconnected via Internet is considered of vital importance in order to minimize financial losses when attacks succeed. Few institutions, however, have adopted such procedures, due to the high cost involved and technical difficulties for deployment. [65]

Preparing for a consistent response starts well before the incident, with the design of procedures and policies, responsibility assignment and personnel training. Procedures to be adopted during an eventual incident must be previously tested and rehearsed due to the great pressure involved when the incident occurs, which may lead to procedure errors that may render evidences useless and may impair the progress of the investigation.

An important stage while designing a response procedure is the correct configuration of the networked computers for an eventual forensic analysis. This is particularly important on Windows platforms, where default configurations leave very few footprints of user activities, which

can delay result findings and consequently delay return to normal activities, aggravating the financial losses.

When a Windows machine is previously configured as a target for forensic analysis, the search for evidence is facilitated by correct utilization of resources that allow the collecting of facts occurred in the recent past of the attacked machine [19]. It can be extremely laborious, however, to configure large networks, especially Windows 2000 networks, which has few mechanisms for administrative task automation.

We have developed the tool PFSAF (*Pre-Forensic Setup Automation Framework*) with the goal of automating some administrative tasks that may significantly speed up a possible forensic analysis, and to provide mechanisms for magnifying footprints of illegal use of resources in Windows 2000 machines. PFSAF is a GPL software aiming at remote setup of W2k machines for future analysis during execution of incident response procedures.

This work aims at presenting relevant aspects of W2k machine configuration towards a future forensic analysis, and also at presenting a new framework for automation of tasks related to this configuration.

6.1.1.1 Related Work

While there are several works in the area of administration of large[1] Windows networks, very few focus on the needs of computational forensics. Furthermore, most tools for automation of administrative tasks in Windows NT/2000 networks are proprietary commercial tools, with high costs, unacceptable by many institutions and incident response teams.

Despite the scarcity of tools satisfying the needs described in this work, two works present similarities with our framework: the set of scripts presented by Harlan Carvey in [8], with similar features in the nucleus of PFSAF, such as remote query of shortcuts in the Startup folder. Those scripts, however, are not scalable to large networks: the administrator would have to edit and execute each script on all machines. The second work is DoIt4Me [1], which is a tool for automation of administrative tools in large Windows NT networks, but which does not cover all requirements presented in this work, such as file integrity.

Tripwire, a tool for file integrity verification, is also a related work. While the Windows version is proprietary, there is an open source project for Linux [60].

1. Networks with several hundred or thousands of machines.

6.1.1.2 Incident Response

Security awareness is nowadays an essential requisite for the majority of network applications. J. Carpenter, a senior security engineer of CERT/CC (Computer Emergency Response Team/Coordination Center), has presented a good argument: “the history of security in the Internet can be compared to life in a town. When the town is small, all people know each other and trust one another and windows and doors can be left open (…). As the town grows, crime and security become issues of concern. The Internet today can be compared to a big metropolis, where doors and windows must be kept closed most of the time” [6].

The main problem to face is: even when all security measures are taken, security failures may occur, because some non disclosed or known vulnerability may be exploited by a new attack. One can not assert, therefore, that no matter what security apparatus we have (Firewalls, VPNs, etc.), our system is immune to attacks. This is due to the fact that such apparatus as services provided through the Internet are composed of many software components with thousands of lines of code not immune to programming errors.

Assuming that there is no security scheme immune to failures, it is necessary to define procedures to be followed in case of a well succeeded attack, besides the availability of people to perform these functions (Response Team). Awareness with such methodologies is still rare within corporations, what can aggravate losses when such events occur.

6.1.1.3 Forensic Discipline

The term “forensic” forwards us immediately to the police environment, where in order do solve a mystery, policemen and experts must minutely analyze all sorts of objects, signals and marks present in the crime scene.

Forensic analysis starts immediately after the arrival of policemen, with an efficient perimeter isolation, avoiding excessive exposition and possible contamination of evidences. One passes then to the stage of identifying and collecting all types of data an material which could have some relevance in the solution of this case. Only after these two phases, one starts with laboratory analysis of possible evidences, such as ballistic and DNA analysis. This demonstrates the mistake of many people, considering only laboratory analysis as forensic analysis… [26].

The success of the analysis is directly connected to the success of all three stages that constitute the process. If there is contamination or faults in the application of methodologies for acqui-

sition or handling of evidence, we have a great possibility of not getting accurate results, or even any result at all during laboratory analysis, applying technical considerations; such failures completely invalidate any evidence in a court.

Required techniques for laboratory analysis of an evidence are divided in phases which depend directly on the type of material being analyzed, that can vary from a corpse to a minuscule paint mark. Take as an example a DNA analysis from a blood sample collected at the crime scene. It is possible to apply the same protocol to every DNA sample received: one eliminates impurities and reduce the sample to its elementary form [41]. All these procedures are standardized, generate reproducible results and are acceptable by the international scientific community.

6.1.2 Computer Forensics

The advent of the computer and the first digital crimes involving the computer environment, made necessary the creation of a new forensic discipline, to act in this new niche, creating methodologies and cumulative knowledge in the acquisition, handling and analysis of digital evidence.

The solution of a computational mystery can be an arduous and difficult task. The system must be meticulously examined, in the same way as a detective examines a crime scene [19]. For this purpose the person making the analysis must thoroughly know the operating system in order to identify and understand the cause-effect relationships of all actions taken during the analysis.

Cause-effect relationships are not sufficient, however; there is the need of more skills in order that a professional can effectively conduct a forensic analysis. Fortunately, according to Venema & Farmer [19], many of these skills are characteristic of programmers, such as logical reasoning and an open mind. Such skills are largely used during the search for the cause of errors in a program. Debugging a faulty program, however, is far away from the challenge of forensic analysis, because when someone debugs a program he his fighting against himself, while in forensic analysis one is challenging another programmer who is not interested in being discovered [19].

6.1.2.1 Digital Evidence Manipulation Standards

In this section we present an overview of the present stage of the computational forensic standardization efforts. The importance of such efforts is due to the need to guarantee the integrity of evidence presented to a court, since once standardized such procedures, it becomes legally

impractical to deny the facts presented assuming that the methodology was correctly used while dealing with the proofs.

Despite the existence of several works in this area, we still note a scarcity of methodologies for the handling of evidence. This can be explained by the fact that there are numerous media types and operating systems, besides many versions of each. All these factors make difficult the definition of standards and methodologies, at least as compared to other forensic disciplines [44].

Nowadays there are some international standards being experimentally used [41]. They were developed by the SWGDE (*Scientific Working Group on Digital Evidence*), which is the American representative in the IOCE (*International Organization on Computer Evidence*). These standards were presented during the International Hi-Tech Crime and Forensic Conference, in London, 4–7th October 1999.

The standards developed by the SWGDE follow a unique principle: all organizations dealing with forensic investigation should keep a high quality level in order to assure the reliability and precision of evidence. This high quality level can be achieved through the elaboration of SOPs (*Standard Operating Procedures*), which contain procedures for all types of known analysis and provide the use of techniques, equipment and materials accepted by the international scientific community [41].

In Brazil, as an example, there is no standardization in progress, however there is ongoing research directed to this area, as can be seen in [44] and [50]. Also, there are some works done upon request of the Federal Police, aimed at non computer professionals such as prosecutors and federal judges.

6.1.3 W2k Forensic Analysis

It is important to configure machines in a way that takes into account a possible future forensics analysis. To fully understand the implications of the latter statement, one needs to know how an investigation on a W2k machine should be done. However, in this section we cite only a few examples of procedures that might be adopted, since each case presents its own peculiar needs, which accounts for the non-viability of defining procedures applicable to all possible situations that can be found during a security incident.

6.1.3.1 Live Analysis on W2k Machines

A live analysis can be defined as one performed on a system victim of a security incident, before any procedure is taken for shutting it down. This kind of analysis is extremely important for an investigation, given that it is the only opportunity to collect a series of volatile information, which otherwise would not be available after a machine restart. Among these are currently active network connections and programs currently running before initiating the investigation.

The biggest problem with this kind of analysis is the lack of control over the machine under analysis, since it can still be under control of an attacker. That means that a whole assortment of programs and libraries may be active, all developed to conceal information and to lure the investigator. For this reason, it is mandatory for the analysis to be performed from programs and libraries run from trusted media (e.g. CD-ROM), so that the examiner may have assurance of the integrity of the binaries being run.

Another possible problem which the investigator is going to face, during the selection of tools that will make up his application CD (Response Kit), is how he is going to find out all library (DLL) dependencies, needed for those programs to run. One way to do that is through the use of *Dependency Walker (depends.exe).* This tool comes with W2k's Resource Kit and analyses all the dependency tree of a given program, giving a break down of functions used by each library. An alternative is *listdlls.exe*[1], developed by Mark Russinovich. This tool is capable of listing all libraries that are currently being used by a process. As such, it is necessary first to execute the tool being analyzed and then to use listdlls.exe to obtain the DLL list. It can also be useful during a live analysis, where it can be used to analyze a suspect process on a broken-in machine.

This strategy of including all libraries needed to run Response Kit programs during a live analysis can minimize the use of code originated from the victim machine. However, this is not enough to guarantee that no DLL from the victim machine will be used. The reason is that it is supposed to be a live analysis, with the machine in the state it is when approached for the analysis, and as such a number of libraries are already loaded in memory. Any program run after this moment that needs an already loaded library will not have the operating system load it again (known-to-be-good binary from CD), but rather use the currently loaded, and possibly compromised, one. This opens a window of opportunity for the production of false results.[2]

1. http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml

The most appropriate solution to avoid access to insecure dynamically-loaded libraries is to eliminate all dynamic access through static compilation of all required tools for the analysis. Nevertheless, due to the commercial focus of the Windows family, very few tools for this system are open source, which pretty much rules out this approach.

Another possible solution would be the removal of all DLLs present in memory prior to the analysis, but the collateral effects that could be caused are hard to predict, since W2k does not present mechanisms to safely perform this. A possible outcome would be a number of GPFs (*General Protection Fail*) on currently executing processes, which could jeopardize the forensic analysis. One must recall that a fundamental principle in such procedure [19] is, as much as possible, not to disturb the system under analysis.

A Windows 2000 live analysis represents a big problem for computer forensics, due to the previously mentioned problems and to others that will be discussed in Section 6.1.3.3.

6.1.3.2 W2k Analysis Basis

Most of the main steps toward a computer forensic analysis can be ported to several operating systems. W2k is no different, with several currently adopted procedures being originated from other platforms, notably Unix.

After a careful live analysis is done, it may be necessary to shut the victim machine down for a *postmortem* analysis, which may be described as an analysis performed under a controlled environment. In this, appropriate procedures should be done to act over the system and collect information from it at a low level. Among objects of interest in such analyses are the following examples:

- File Slack: Microsoft's operating systems store their files on disk using fixed-size data blocks called *clusters* (Figura 6.1), however files' contents have no restriction in size. As such, only rarely will a file size be a multiple of a *cluster's* size, which prevents optimal storing. Usually, then, the last *cluster* associated to a file is not fully utilized (Figura 6.1), which allows for data excluded from this and past files to be later captured and analyzed.
- RAM Slack: Besides data from previous files resident on disk, the *file slack space* can also hold sets of bytes randomly selected from RAM memory. This happens because Windows (as most other operating systems do) normally writes on disk in 512-byte blocks, called

sectors. Since normally the amount of information to be written is not a multiple of 512, usually the last block of information will have to be padded to match a sector size (Figura 6.1). Windows uses its own memory buffers to get irrelevant bytes to do that.

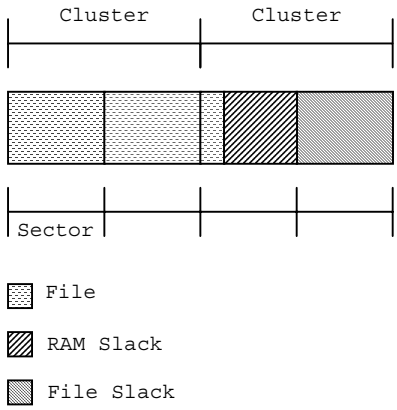


Figura 6.1: Slack Space

- Alternate Streams: it can be said that every NTFS file holds another embedded file with no explicit name, called default stream or unnamed stream, where conventional data like text and programs are stored. Nevertheless, one can also create embedded files with different names, called alternate streams. The problem is that the detection of these embedded, but named, files cannot be natively effected by any W2k programs, enabling them to be easily used to conceal programs and information. [45]

The *postmortem* analysis may be extremely lengthy and laborious. It must be conducted over image copies of the original disks from the victim machine, so that there is no risk of a mistake on the part of the examiner to compromise the integrity of the original files, causing irreparable damage to the ongoing investigation. The controlled environment mentioned previously could be, for instance, a machine with several operating systems containing adequate tools for this sort of analysis, such as the Foundstone Forensic ToolKit[1].

By using another machine with a Unix-like operating system that has support for the NTFS file system, such as Linux, one can take advantage of powerful tools developed for the Unix platform, such as TCT[2] (*The Coroners ToolKit*). Also, this brings the extra advantage of avoiding unintentional changes to the filesystem (the legal evidence), since the media can be mounted read-

1. http://www.foundstone.com/knowledge/proddesc/forensic-toolkit.html
2. http://www.porcupine.org/forensics/tct.html

only, whereas W2k would normally alter the index files of the partition during boot. Another advantage of using a platform like Linux for the *postmortem* analysis is the possibility of visualizing the evidence from another point of view, which might circumvent a possible bias of the view provided by Windows.

Unfortunately, Linux's support for the NTFS filesystem is not complete, so that several data structures, such as the alternate streams, are ignored. This makes it impossible to conduct all the analysis under this system and forcing at least part of the procedures to be performed under W2k itself.

As a general rule, one can say that the majority of conclusions from a forensic analysis is based on results obtained from a live analysis, seconded by an analysis of events registered by EventViewer and trailed by the file integrity checking. Not always a detailed postmortem over the filesystem is performed, or even needed. Often, it is not viable to create disk images due to size constraints on storage available in the analysis machine, or even due to the impact the analysis might impose on running services, such as stopping them altogether, which may not be acceptable [28].

6.1.3.3 W2k Forensic Analysis Problems

The forensic analysis of a proprietary system such as Windows 2000 makes it harder to define fully reliable methodologies, once it is not known for sure the exact consequences of all actions taken during its analysis [45].

Since W2k is a undisclosed-code system, it is not possible to prove with full certainty that, during evidence handling, no result contamination may have happened due to an undocumented procedure or data manipulation by Windows. On the other hand, the availability of source code entitles us not to blindly trust the maker's documentation, as the accuracy of all information provided can be confirmed by an analysis of such code.

The availability of said code allows the legal expert to certify his methodology's correctness, turning down any discrediting attempt at his analysis on court. This comes from the possibility to prove that no applied procedure performed any significant alterations in the original state of the system, through source code analysis.

Another problem worth mentioning is that in the Windows environment the GUI culture is praised, which eases system usage but also tends to hide location and treatment given to infor-

mation being handled. This rarely happens on open source systems like Linux and FreeBSD when documentation is not available. The lack of such information makes it difficult gathering data from the target system without the use of tools originally available on the operating system. Sometimes the analysis is further impaired due to the need to manipulate data in unconventional ways [19].

6.1.4 Pre-Forensic Setup

Despite current security concerns being essential requirement for a variety of network services and applications, it is still very common to find W2k networks with out-of-the-box configurations. These usually deal only with functional aspects, while security configurations are given a lower priority.

One of the main reasons for W2k machines not providing adequate information during forensic analyses is the fact that most administrators use default Windows configurations, which generally speaking tend not to deal with security issues properly, if at all. Take, for instance, the auditing policy configurations, which are not enabled by default, leaving all security-related events not to be logged.

Event Category	Description
Account logon events	Activated when a domain controller receives a logon request
Account management	Activated when a user account or group is created or changed
Directory service access	Activated when an Active Directory object is accessed
Logon events	Activated when a user logs on or logs off
Object access	Activated when an object is accessed
Policy change	Activated when a policy affecting security, user rights, or auditing is modified
Privilege use	Activated when a user right is used to perform an action

Tabela 6.1: Auditing categories

Event Category	Description
Process tracking	Activated when an application executes an action that is being tracked
System events	Activated when a computer is rebooted or shut down or another event occurs that affects security

Tabela 6.1: Auditing categories

The correct configuration of a machine, where a possible, future forensic analysis is taken into account, can substantially speed up the response to a security incident, and also possibly bring its solution closer. This is due to the fact that a correct configuration will force W2k to supply many more traces of an attacker's actions, further enlarging the collection of evidences that could draw the case to an end.

Below are discussed some examples of topics whose configuration or deployment could be of great benefit for the progress of an analysis:

- Audit Policies: Event auditing is an important part of network administration. When an administrator selects events to be audited, he can later, through the analysis of log files, derive normal system utilization patterns, possible security problems and observe trends in network resource utilization. Care, however, should be taken when choosing what is to be audited, given the large number of events some activities can generate and the overhead associated with them [35][34]. Tabela 6.1 shows a description of all auditable events on W2k and Tabela 6.2 shows some interpretations for the logs they generate.

- ACLs: NTFS stores an access control list (ACL) with every file and folder on an NTFS volume. The ACL contains a list of all user accounts and groups that have been granted access for the file or folder, as well as the type of access that they have been granted. When a user attempts to gain access to a resource, the ACL must contain an entry, called an access control entry (ACE), for the user account or a group to which the user belongs. The entry must allow the type of access that is requested (for example, Read access) for the user to gain access. If no ACE exists in the ACL, the user cannot gain access to the resource. The ACL setup with a correct audit policy configuration, can show general tracks of system utilization.[34]

Audit Event	Potential Threat
Failure audit for logon/logoff	Random password hack
Success audit for logon/logoff	Stolen password break-in
Success audit for user rights, user and group management, security change policies, restart, shutdown, and system events	Misuse of privileges
Success and failure audit for file-access and object-access events. File Manager success and failure audit of Read/Write access by suspect users or groups for the sensitive files.	Improper access to sensitive files
Success and failure audit for file-access printers and object-access events. Print Manager success and failure audit of print access by suspect users or groups for the printers.	Improper access to printers
Success and failure write access auditing for program files (.EXE and .DLL extensions). Success and failure auditing for process tracking. Run suspect programs; examine security log for unexpected attempts to modify program files or create unexpected processes. Run only when actively monitoring the system log.	Virus outbreak

Tabela 6.2: Potential Threats[35]

- File Integrity Check: It is very difficult to compromise a system without altering a system file, so file integrity checkers are an important capability in the search for evidences and tracks. A file integrity checker computes a checksum for every critical file and stores this. At a later time you can compute a checksum again and test the current value against the stored value to determine if the file has been modified.[35][60]

6.1.5 Pre-Forensic Setup Automation Framework

The dissemination of corporate networks and their ever growing size made the creation of techniques for large network administration a very promising research field. Countless techniques were developed for the Unix world, which was the sole mainstream platform in corporate networks till recently. Since the inception of Windows NT 4.0, the Windows platform has been gaining ground in this area and today Windows 2000 networks of hundreds and even thousands of machines are common.

Contrary to Unix systems, Windows is considered a “hands-on” system, which requires the administrator’s presence or actuation on each machine, so that most installations, configurations and administrative tasks are performed on a network. This makes the administration of a large Windows network an extremely laborious and unproductive task.

The administrative hardships imposed by the Windows platform are a severe obstacle for the deployment of the recommendations on Section 6.1.4. They, in fact, cause such deployment to take quite a long time from the Response Team. In this section we present a framework implementation (PFSAF) that seeks to automate the main actions required to prepare machines on a network for the chance of a future forensic analysis.

6.1.5.1 Structure

PFSAF was structured so as to allow all necessary configurations from a single machine, but also being able to schedule tasks through Task Scheduler at a more convenient time for the administrator. This machine (Forensic Station) must be of restricted access and will be responsible for the storage of all configuration files and databases generated by PFSAF (Figura 6.2).

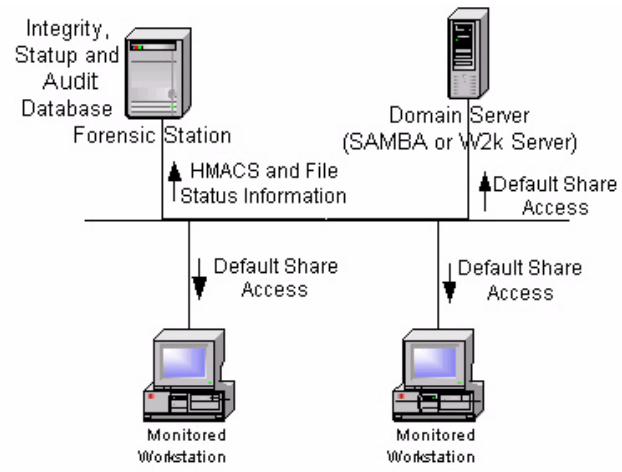


Figura 6.2: File Integrity Module Schema

The machines to be monitored do not need any special configuration to comply with PFSAF; the only requirement is the existence of the W2k default shares (Figura 6.5).

Currently the main functionalities of PFSAF are:

- **Guarantee of critical files integrity:** the PFSAF file integrity module generates cryptographic hashes, registers the existence and size of alternate streams and stores the extremely volatile MAC Times (access, modification and creation times of a file);

- **Automatic execution program monitoring:** all programs automatically executed after user logon has its integrity monitored. Moreover, any change in the number of programs to be executed or alterations in their characteristics can be detected;
- **Auditing policy configuration:** PFSAF can remotely configure auditing policies and check whether they have been changed afterwards.

6.1.5.2 Implementation

PFSAF was implemented using PERL, which is extremely versatile on file and string handling (the main need of this implementation) and also possesses several libraries[1] devoted to Windows machine administration, even though was born in the Unix environment. Also, it has extensive documentation and is distributed under the GNU GPL license.

```
c:\boot.ini
c:\io.sys
c:\config.sys
c:\autoexec.bat
c:\winnt\system32\fpnwclnt.dll
```

Figura 6.3: File `files.cfg`

```
#Event Order:
#
#server(Process Tracking,
#       Account Logon,
#       Restart and Shutdown,
#       User/Group Management,
#       Dir. Service Access,
#       File and Object Access,
#       Security Policy Changes,
#       Logon and Logoff,
#       Use of User Rights)
#
#Audit Values:
#
#        None                -> 0
#        Success             -> 1
#        Failure             -> 2
#        Success and Failure -> 3
#        Unchanged           -> 4
#
#Example:
#
#testserver(2,0,0,4,0,0,0,3,0)

obiwan(0,0,0,0,0,0,0,3,0)
jaba(0,0,0,0,0,0,0,2,0)
anakin(0,2,0,0,0,0,0,3,0)
```

Figura 6.4: File `audit.cfg`

1. For the sake of notation, PERL language extensions, usually called modules, will be referred here as libraries in order not to be confused with PFSAF's blocks, which also are designated modules.

PFSAF was tested with the ActiveState[1] 5.6.1 Build 631 interpreter on a Windows 2000 Server with Service Pack 2, having Windows 2000 Server and Professional clients on a variety of Service Pack combinations.

There are basically two configuration files in this framework: `files.cfg` (Example 6.3) and `audity.cfg` (Example 6.4). The file `files.cfg` holds filenames to be included in the integrity database (`hash.db`), which is responsible for the storage of cryptographic hashes and some extra data which describe the structure and the state of the target file.

File `files.cfg` is used for all machines listed in file `audit.cfg`, unless there exists a specific file for a given machine, whose name has the following syntax: `<machine>-files.cfg`. In this case, file `files.cfg` is neglected under the more specific configuration listing.

Names of the machines to be monitored are stored in file `audit.cfg`, which also describes which auditing policies must be applied on each listed machine.

In this section will be discussed implementation details of each module presented in Section 6.1.5.1.

File Integrity Module

The file integrity module uses both PFSAF configuration files: `files.cfg`, which is specific to this module and `audit.cfg`, which holds the list of machines to be monitored, besides the description of auditing policies for each machine.

Remote access to each machines' files is done through Windows *default shares* (Figura 6.5), which, though criticized by some, have become extremely useful for system administration.

For cryptographic hashes, PFSAF uses SHA1 paired with the HMAC symmetric key primitive [29], also called HMAC-SHA1. Its use prevents a simple cut-and-paste attack to the `hash.db` file, in which the attacker could substitute a compromised hash for the correct one, therefore validating a possible change in the monitored files.

HMAC-SHA1 uses a key to compute the file hashes, so that it is not possible for anyone not possessing that key to produce any valid hashes for substitution over the `hash.db` file. In the case of PFSAF, a key is required when generating the database and stored in the `passwd` file (also part of PFSAF). The HMAC-SHA1 support in PERL is given by the Digest::HMAC_SHA1 library.

1. http://www.activestate.com/Products/ActivePerl/

Startup Integrity Database

The Startup Integrity Module is responsible for monitoring and ensuring the integrity of all automatically executed programs after user logon. Its goal is to prevent a maliciously altered program or backdoor from being automatically executed in any of the monitored machines.

There are two ways for a program to automatically execute when a user logs on a Windows machine.

The simplest one is through the addition of shortcuts in the Startup folder, present in all user profiles. To deal with this case, PFSAF accesses the folder holding all user profiles, again by using Windows default shares. The shortcut analyses are made easier thanks to Win32::Shortcut, a very helpful PERL library. Once the executable pointed to by the shortcut is known, PFSAF adds two registers to the Startup Database: one holds the hash and the state of the shortcut file and the other the executable's.

```
C:\>net share
Share name   Resource                        Remark
------------------------------------------------------------
IPC$                                         Remote IPC
D$           D:\                             Default share
G$           G:\
ADMIN$       C:\WINNT                        Remote Admin
C$           C:\                             Default share
The command completed successfully.
```

Figura 6.5: Default Shares on W2k Machine

The other one is through the addition of proper values in some register keys, which are:

- HKEY_LOCAL_MACHINE\SOFTWARE\
  Microsoft\Windows\CurrentVersion\Run: Programs included in this key are simultaneously executed after any user login;
- HKEY_LOCAL_MACHINE\SOFTWARE\
  Microsoft\Windows\CurrentVersion\RunOnce: Programs included in this key are sequentially executed, following the order they were added (not necessarily the one shown by the registry editor);
- HKEY_LOCAL_MACHINE\SOFTWARE\
  Microsoft\Windows\CurrentVersion\RunOnceEx: Used by Windows Desktop Update, this key is an attempt to make the automatic execution of programs more robust, as it features mechanisms for error handling and documentation as well as ways to increase application performance. More information can be obtained in [31];

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run: Unlike LOCAL_MACHINE's Run key association to all users, this key holds exclusive configurations for each individual user who logs in. Such information is instantiated during logon from the corresponding one in the HKEY_USERS hierarchy;
- HKEY_CURRENT_USER\Software\Microsoft\ Windows\CurrentVersion\Runonce: Similar to its cousin in the LOCAL_MACHINE hierarchy; however, holds exclusive configurations for each user.

PFSAF also features a parser to analyze calls to native W2k script interpreters, such as `wscript.exe` and `cmd.exe`, that may be present on both the key values above and the Startup folder shortcuts.

The command line for a script execution can be rather complex, due to the large number of options offered by interpreters. Many options also have mandatory parameters that vary from interpreter to interpreter. This parser is used in an attempt to find the script filename among all possible options, so that it may be added to the integrity database. The supported interpreters are:

- wscript e cscript: Both are part of Microsoft Windows Script Host (WSH). WSH is a language-independent scripting host for Windows Script compatible scripting engines. It brings simple, powerful, and flexible scripting to the Windows 32-bit platform, allowing one to run scripts from both the Windows desktop (wscript) and the command prompt (cscript).
- cmd: it is the standard W2k command interpreter, used to run batch-type scripts, common to all Microsoft operating systems.
- rundll: `rundll32.exe` allows the invocation of a function exported from a 32-bit DLL. However, Rundll32 does not allow calling any exported function from any DLL. For instance, this utility cannot be used to call the Win32 API (Application Programming Interface) calls exported from the system DLLs. This program only allows calling functions from a DLL that are explicitly written to be called this way. [36]

Audit Policy Module

The goal of the audit policy module is to supply a new interface for auditing policies configuration, thus allowing that either PDCs, which do not share domain configurations, isolated machines or Unix servers running SAMBA may be quickly configured in an automated way.

Through the use of `audit.cfg` (Example 6.4), the audit policy module can configure machines listed in it with their corresponding policy. This module can also check the current policy state on each machine and verify that the comply with current values in `audit.cfg`.

Access to this kind of configuration is granted by PERL using the Win32::Lanman library, that works as a wrapper for the Windows LANMAN API.

6.1.5.3 Usage

PFSAF can be used either interactively or automatically. In the interactive option, the program uses the `main.pl` script (Appendix 6.1.7) that provides an interface for using the whole framework (Figura 6.6), with configuration files prepared beforehand.

```
C:\> main.pl

        [1] Generate Integrity File Database.
        [2] Check Integrity File Database.
        [3] Set Audit Policy.
        [4] Check Audit Policy.
        [5] Query Audit Policy.
        [6] Generate Startup Database.
        [7] Check Startup Database.
        [8] Query Startup Database.
        [9] Help.
        [10] Quit.

                Choose an Option:
```

Figura 6.6: Options available on `main.pl`.

The non-interactive use of PFSAF can be done through calling individual modules (Appendix 6.1.7), separately developed to supply a flexible interface to the framework and so allowing the execution of independent modules through Windows Task Scheduler.

6.1.5.4 Future Work

PFSAF, though fully functional, is far from finished as regarding several functionalities that we desire to aggregate towards easing the work of Response Teams and administrators. Among which are ACL remote configuration and a possible merge of functions offered by DoIt4Me 6.1.7, by porting part of its code do run under W2k.

Another possible feature that could be created is an interface for clock synchronization of monitored machines, which would render MAC Times analysis much friendlier.

Some existing functions also deserve improvements towards a more flexible PFSAF. One example is a more comprehensible syntax for the configuration files that would allow similar specifications to be condensed by using metacharacters.

6.1.6 Conclusion

The use of tools like PFSAF can promote the creation of incident response programs by institutions, since their complex deployment on Windows 2000 based networks can be greatly reduced through task automation. Also, it makes basic requirements for this kind of program more apparent, consequently contributing to spread out the necessity of deploying such programs.

6.1.7 Appendix A - Files

- `audit.cfg:` file that holds the list of machines to be monitored by PFSAF and their corresponding auditing policies (Example 6.4);
- `files.cfg:` list of files to be monitored by the file integrity module. This file is used for all machines listed in `audit.cfg,` unless there is a specific configuration for a given machine in a file whose name follows this syntax: `<machine>-files.cfg` (Example 6.3);
- `main.pl:` script responsible for PFSAF's interactive interface. It prints a menu of options and executes the appropriate scripts for the choice;
- `genintdb.pl:` responsible for the file integrity database generation, it remotely accesses machines listed in `audit.cfg` and collects information from files listed in `files.cfg;`
- `chkintdb.pl:` checks the file integrity database and produces a report;
- `adscount.pl:` auxiliary script for counting alternate streams on a file and for calculating the file's total size;
- `genstartdb.pl:` accesses the Startup folder of all users and the registry keys used for automatic program execution, analyses all entries and generates a database with the collected information;
- `chkstartdb.pl:` checks information in the startup database and reports possible occurrences;

- `keytime.pl:` auxiliary script for registry key information collection, such as number of values and date of last change;
- `parser.pl:` responsible for analyzing command lines that involve calling native W2k script interpreters;
- `auditpol.pl:` configures auditing policies held by `audit.cfg` on corresponding machines;
- `chkaudit.pl:` checks whether the auditing policies on the monitored machines match the ones in `audit.cfg`;
- `hash.db:` file integrity database;
- `start.db:` startup database;
- `passwd:` secret key used by PFSAF during cryptographic hash generation;

6.1.8 Appendix B - Snapshots

- Example 6.7 shows the output of the `genintdb.pl` script when generating a new file integrity database.
- Example 6.8 shows part of the output of the `genstartdb.pl` script while applying it over a typical machine.

```
        Generating File Integrity Database....

Base Directory        : C:\Temp\Images\MyTripwire
Default File Policy   : files.cfg
Hash file             : hashs.db
Audit cfg             : audit.cfg
Startup file          : start.db

Hash files.cfg: 0e1c55162fb624719984f3cb0b1482f43bf944c6
Hash audit.cfg: 8ee665effaed40ede378be90ae8f67c0ed45d140
-----------------------------------------------------------------
Machine: ANAKIN
-> Using default file policy!

Can't open c:\Temp\images\MyTripwire\test.txt...
Hash c:\boot.ini: 096cf25dcebe81df1fa46ea584fa44d6fb431e48
Hash c:\io.sys: da39a3ee5e6b4b0d3255bfef95601890afd80709
Hash c:\config.sys: da39a3ee5e6b4b0d3255bfef95601890afd80709
Hash c:\autoexec.bat: da39a3ee5e6b4b0d3255bfef95601890afd80709
Hash c:\winnt\system32\fpnwclnt.dll: 91754c5917f1805069225089ea4d9d73ee1d154f

There were files NOT processed in files.cfg!!!
```

Figura 6.7: Using genintdb.pl to generate a new hash.db.

```
C:\Temp\Images\MyTripwire>genstartdb.pl

                Generating Startup Integrity Database

Machine List   : audit.cfg
Startup Database: start.db
-----------------------------------------------------------
Machine: anakin

Checking Key: LMachine/SOFTWARE/Microsoft/Windows/CurrentVersion/Run

> C:\WINNT\SYSTEM32\QTTASK.EXE -> 222a694f6044bf99988f2804ea6b6eccda15cd14
> C:\PROGRAM FILES\WINAMP\WINAMPA.EXE -> 21baf13bcad02df06cf2b87e9164a7a369bf471
4
> C:\PROGRAM FILES\AHEAD\INCD\INCD.EXE -> a6e3a43b9f8cca7383d025017aa993f8a233e4
38
> C:\WINNT\system32\NVQTWK.DLL -> abe578de7fa3899a3d385f0f0740f98976cdea5f
> C:\PROGRA~1\ADAPTEC\EASYCD~1\CREATECD\CREATECD.EXE -> 19814146771840b3f1c1e497
789c4b228ccbb30d

Checking Key: LMachine/SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnce
Checking Key: LMachine/SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnceEx
Checking Key: LMachine/SOFTWARE/Microsoft/Windows/CurrentVersion/RunServices
Checking Key: Users/.Default/SOFTWARE/Microsoft/Windows/CurrentVersion/Run

> C:\WINNT\system32\INTERNAT.EXE -> 08f1b7c33c0f5c3ed7aad0752bd63d96db1d4c2f

Checking Key: Users/.Default/SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnce
Checking Key: Users/.Default/SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnceEx
Checking Key: Users/.Default/SOFTWARE/Microsoft/Windows/CurrentVersion/RunServices

- Checking Startup Directories

Cheking: C:\Documents and Settings\Administrator\start menu\programs\startup
Cheking: C:\Documents and Settings\Administrator.ANAKIN\start menu\programs\star
tup
> C:\WINNT\SYSTEM32\CMD.EXE -> c653d5b88cdd2389ea27a54d2eefe78e741c8f7d
Cheking: C:\Documents and Settings\All Users\start menu\programs\startup
> C:\PROGRAM FILES\GETRIGHT\GETRIGHT.EXE -> 508fdd2fa7bb1162bab86f2d5e4d9156dd91
1828
```

Figura 6.8: Output produced by executing the `genstartdb.pl` script.

6.2 Resumo

No decorrer deste capítulo foram apresentados algumas bases da forense computacional seguidas de problemas enfrentados durante às análises forenses em máquinas Windows, que não estejam previamente preparadas para este tipo de procedimento. Foram descritos alguns elementos cuja configuração proveria mais evidencias durante uma análise forense e as dificuldades de implantação dessas configurações devido aos problemas administrativos inerentes ao Windows 2000 especificamente. Tal cenário motivou o desenvolvimento do PFSAF, cujas características, estrutura e implementação compõem o restante do capítulo.

Capítulo 7

Conclusões

A implantação de um programa de resposta a incidentes em uma organização pode ser uma tarefa bastante complexa, por ter impacto nas políticas de utilização de recursos de TI da instituição, o que por sua vez pode afetar um grande número de usuários. Além disso, existem diversas dificuldades técnicas a serem enfrentadas, desde a formação de um Time de Resposta, que deve contar com membros dotados de características técnicas bem específicas, até a definição de todos os passos a serem seguidos durante uma análise forense.

Apesar de não haver inicialmente qualquer relação entre a criação de programas de resposta a incidentes e o tamanho da organização envolvida, a abrangência dos programas pode ser intrinsecamente afetada por este aspecto. Dado que, grandes organizações podem contar com uma maior quantidade de mão de obra, permitindo a implantação de programas mais amplos, isto é, programas que possuem procedimentos para lidar com grande parte das etapas de resposta sem a necessidade de acionar um agente externo. Entretanto, este fato não exime pequenas instituições da necessidade deste tipo de programa.

O presente trabalho buscou apresentar aspectos estruturais dos programas de resposta a incidentes, agrupar e discutir metodologias de análise forense para Windows 2000, que é um sistema carente de estudos deste tipo. Durante tal estudo foi possível mapear as dificuldades enfrentadas durante as análises de redes baseadas nesta plataforma, o que motivou a implementação do PFSAF, com o intuito de automatizar tarefas administrativas que podem ampliar a quantidade de evidências deixadas por uma utilização ilegal de determinada máquina.

As técnicas discutidas neste trabalho vêm de encontro às necessidades fundamentais deste tipo de projeto e podem ser de grande valia para programas de resposta a incidentes de variadas

amplitudes. No entanto, o PFSAF, apesar de poder contribuir também para pequenas organizações, tem o seu potencial melhor utilizado quando o TR deve lidar com grandes redes Windows 2000 devido às dificuldades de centralização de tarefas e administração remota dessas redes.

7.1 Trabalhos Futuros

Como observado na Seção 6.1.5.4 o PFSAF ainda pode agregar funções, que podem aumentar sua atual capacidade de atuação. Seriam úteis também, aprimoramentos em sua interface interativa e melhoramentos na sintaxe de seus arquivos de configuração, o que implica na construção de um novo *parser* que possa interpretar metacaracteres e expressões mais complexas.

O trabalho em torno de técnicas para análise forense é contínuo, uma vez que novos programas e versões de sistema operacional surgem constantemente; logo, técnicas muito específicas tendem a se desatualizar rapidamente. Outro ponto a ser analisado é o possível surgimento de padrões internacionais e legislação especifica para o assunto, que podem influir significativamente na maneira como as máquinas são preparadas para uma futura análise.

Uma próxima fase de desenvolvimento deste projeto seria a realização de testes em uma rede formada por estações Windows XP, que é o próximo sistema operacional na linha de evolução da Microsoft. Embora haja grande compatibilidade entre as redes W2k e XP, certamente existe a necessidade de ajustes na implementação, o que aumentaria consideravelmente a vida útil do *framework*.

Apêndice A

Exemplo de SOP

O contexto do presente SOP é o da análise *postmortem* de uma máquina vítima de um incidente de segurança. Após o desligamento da máquina em questão existe a necessidade de se efetuar cópias bit-a-bit de seu disco rígido para que os procedimentos de análise não sejam conduzidos a partir das evidências originais, evitando-se assim, o risco de que um eventual erro do examinador venha a danificar ou alterar a evidência original de alguma forma.

Este exemplo foi baseado no procedimento descrito em [50].

Empresa X S/A

Time de Resposta a Incidentes Segurança - TRIS

Documento: AF/0013

Responsável:

Flávio de Souza Oliveira
flavio.oliveira@ic.unicamp.br
Ramal: 85857

Versão: 1.0 - 03/10/2002

Descrição:

Procedimento para duplicação bit-a-bit de disco rígido IDE proveniente de máquina envolvida em incidente de segurança.

Requisitos:

Máquina confiável utilizando Linux Red Hat 7.3 (AF/0008) com quantidade de espaço livre em disco superior à capacidade total de armazenamento do disco que se deseja duplicar. Presença dos programas `dd` e `md5sum` em conjunto com suas bibliotecas certificadamente originais (AF/0010).

Procedimento:

1. Documentação de todas as informações relevantes impressas na superfície externa do disco, tais como modelo, fabricante, quantidade de cilindros e cabeças de leitura. Além do registro das posições dos *jumpers* presentes no equipamento;

2. O disco deve ser instalado na estação forense no canal IDE secundário, e para que não haja conflitos em relação às configurações de dominância (*master/slave*), o disco deve ser o único dispositivo presente no canal. Para os próximos passos será suposto o mapeamento do disco alvo no dispositivo de bloco `/dev/hdc`;

3. Ligue a estação e execute a detecção de discos presente na BIOS, tomando o cuidado de documentar a geometria do disco detectado;

4. Identifique e documente as partições presentes no dispositivo através do comando `fdisk -l /dev/hdc`;

5. Calcule o *hash* MD5 de todos os dados contidos no disco através do comando `dd if=/dev/hdc | md5sum -b`;

6. Efetue a cópia de cada bit presente no dispositivo analisado, incluindo os espaços aparentemente vazios. Isto pode ser feito através do comando `dd if=/dev/hdc of=<nome-do-arquivo>`. Note que tal procedimento pode exigir uma considerável capacidade de disco da estação forense.

7. Calcule o *hash* MD5 do arquivo gerado no passo 6 através do comando `dd if=<nome-do-arquivo> | md5sum -b`;

8. Verifique se os *hashes* criptográficos gerados nos passos 5 e 7 correspondem exatamente ao mesmo valor;

Apêndice B

Aspectos de Implementação e Utilização do PFSAF

Este apêndice trata de detalhes da implementação e utilização do PFSAF, conta com exemplos de arquivos de configuração e discorre sobre sua utilização. O apêndice traz também alguns trechos de código do *framework* propriamente dito, dado que a inclusão de todo o código fonte é inviável e desinteressante devido à sua extensão.

B.1 Arquivos de Configuração

Nesta seção se fará uma breve descrição dos dois arquivos de configuração utilizados pelo PFSAF, seguidas de alguns exemplos.

B.1.1 Files.cfg

O arquivo `files.cfg` não passa de uma lista de arquivos críticos a serem monitorados. Este arquivo geral é utilizado para todas as máquinas, a não ser que haja algum arquivo mais específico para determinada máquina, cujo nome esteja no formato `<máquina>-files.cfg`.

A lista de arquivos pode ser baseada nos arquivos gerados pelo gerênciador de políticas presentes no Tipwire Policy Center[1], e posteriormente adaptado para o formato utilizado pelo PFSAF e para as necessidades peculiares de cada organização.

1. http://tpt.tripwire.com/tprc/servlet/OsChoose

B.1.1.1 Trecho de um arquivo files.cfg

```
C:\WinNT\NOTEPAD.EXE
C:\WinNT\Speech\vcmd.exe
C:\WinNT\TASKMAN.EXE
C:\WinNT\delttsul.exe
C:\WinNT\explorer.exe
C:\WinNT\hh.exe
C:\WinNT\inf\unregmp2.exe
C:\WinNT\msagent\agentsvr.exe
C:\WinNT\poledit.exe
C:\WinNT\regedit.exe
C:\WinNT\System32\AUTOCHK.EXE
C:\WinNT\System32\AUTOCONV.EXE
C:\WinNT\System32\CACLS.EXE
C:\WinNT\System32\CHKDSK.EXE
C:\WinNT\System32\CHKNTFS.EXE
C:\WinNT\System32\CLUSTER.EXE
C:\WinNT\System32\CMD.EXE
C:\WinNT\System32\CONVERT.EXE
C:\WinNT\System32\CSRSS.EXE
C:\WinNT\System32\Com\comrepl.exe
C:\WinNT\System32\Com\comrereg.exe
C:\WinNT\System32\DCOMCNFG.EXE
C:\WinNT\System32\DLLHOST.EXE
C:\WinNT\System32\DRWTSN32.EXE
C:\WinNT\System32\FTP.EXE
C:\WinNT\System32\LABEL.EXE
C:\WinNT\System32\LLSSRV.EXE
C:\WinNT\System32\LOCATOR.EXE
C:\WinNT\System32\LSASS.EXE
C:\WinNT\System32\NETDDE.EXE
C:\WinNT\System32\NSLOOKUP.EXE
C:\WinNT\System32\NTBACKUP.EXE
C:\WinNT\System32\NTKRNLPA.EXE
C:\WinNT\System32\NTOSKRNL.EXE
C:\WinNT\System32\NTVDM.EXE
C:\WinNT\System32\ODBCAD32.exe
C:\WinNT\System32\Prounstl.exe
C:\WinNT\System32\RECOVER.EXE
C:\WinNT\System32\SAVEDUMP.EXE
C:\WinNT\System32\SERVICES.EXE
C:\WinNT\System32\SMSS.EXE
C:\WinNT\System32\SPOOLSV.EXE
C:\WinNT\System32\USERINIT.EXE
C:\WinNT\System32\WINLOGON.EXE
C:\WinNT\System32\accwiz.exe
C:\WinNT\System32\acsetups.exe
C:\WinNT\System32\actmovie.exe
C:\WinNT\System32\append.exe
C:\WinNT\System32\arp.exe
C:\WinNT\System32\at.exe
C:\WinNT\System32\ati2evxx.exe
C:\WinNT\System32\atiiprxx.exe
C:\WinNT\System32\atiphexx.exe
C:\WinNT\System32\atiptaxx.exe
C:\WinNT\System32\atmadm.exe
C:\WinNT\System32\attrib.exe
C:\WinNT\System32\autofmt.exe
C:\WinNT\System32\autolfn.exe
C:\WinNT\System32\bootok.exe
C:\WinNT\System32\bootvrfy.exe
C:\WinNT\System32\calc.exe
C:\WinNT\System32\cdplayer.exe
C:\WinNT\System32\change.exe
C:\WinNT\System32\charmap.exe
C:\WinNT\System32\chglogon.exe
C:\WinNT\System32\chgport.exe
C:\WinNT\System32\chgusr.exe
C:\WinNT\System32\cidaemon.exe
C:\WinNT\System32\cipher.exe
C:\WinNT\System32\cisvc.exe
C:\WinNT\System32\ckcnv.exe
C:\WinNT\System32\cleanmgr.exe
C:\WinNT\System32\cliconfg.exe
C:\WinNT\System32\clients\clcreate.exe
C:\WinNT\System32\clipbrd.exe
C:\WinNT\System32\clipsrv.exe
C:\WinNT\System32\clspack.exe
C:\WinNT\System32\cmdl32.exe
C:\WinNT\System32\cmmgr32.exe
C:\WinNT\System32\cmmon32.exe
C:\WinNT\System32\cmstp.exe
C:\WinNT\System32\comclust.exe
C:\WinNT\System32\comp.exe
C:\WinNT\System32\compact.exe
C:\WinNT\System32\conime.exe
C:\WinNT\System32\control.exe
C:\WinNT\System32\convlog.exe
C:\WinNT\System32\cprofile.exe
C:\WinNT\System32\cscript.exe
C:\WinNT\System32\csvde.exe
C:\WinNT\System32\dbgtrace.exe
C:\WinNT\System32\dcphelp.exe
C:\WinNT\System32\dcpromo.exe
C:\WinNT\System32\ddeshare.exe
C:\WinNT\System32\ddmprxy.exe
C:\WinNT\System32\debug.exe
C:\WinNT\System32\dfrgfat.exe
C:\WinNT\System32\dfrgntfs.exe
C:\WinNT\System32\dfscmd.exe
C:\WinNT\System32\dfsinit.exe
C:\WinNT\System32\dfssvc.exe
C:\WinNT\System32\diantz.exe
C:\WinNT\System32\diskperf.exe
C:\WinNT\System32\dllhst3g.exe
C:\WinNT\System32\dmadmin.exe
C:\WinNT\System32\dmremote.exe
C:\WinNT\System32\doskey.exe
```

B.1.2 Audit.cfg

O arquivo `audit.cfg` possui a lista de máquinas que serão monitoradas em conjunto com sua respectiva política de auditoria, obedecendo o seguinte formato: `máquina(política de auditoria)`, como pode ser observado na Seção B.2.1.

B.1.2.1 audit.cfg exemplo

```
#Event Order:
#
#server(Process Tracking,
#       Account Logon,
#       Restart and Shutdown,
#       User/Group Management,
#       Dir. Service Access,
#       File and Object Access,
#       Security Policy Changes,
#       Logon and Logoff,
#       Use of User Rights)
#
#Audit Values:
#
#        None                -> 0
#        Success             -> 1
#        Failure             -> 2
#        Success and Failure -> 3
#        Unchanged           -> 4
#
#Example:
#
#testserver(2,0,0,4,0,0,0,3,0)

OBIWAN(0,0,0,0,0,0,0,3,0)
JABA(0,0,0,0,0,0,0,2,0)
ANAKIN(0,2,0,0,0,0,0,3,0)
```

B.2 Bases de Dados

O PFSAF utiliza duas bases de dados, uma referente à integridade dos arquivos indicados pelo arquivo de configuração `files.cfg` denominada `hashs.db` e a outra armazena informações a respeito dos programas que são iniciados automaticamente nas maquinas monitoradas, chamada `start.db`.

B.2.1 Hashs.db

O `hashs.db` é o arquivo criado para armazenar informações sobre o estado dos arquivos monitorados em conjuto com informações sobre suas *alternate streams*, caso existam, e seu hash criptográfico.

B.2.1.1 Dicionário de Dados

Segue abaixo os campos existentes em um registro normal do `hashs.db`:

1. Arquivo: nome do arquivo analisado;
2. Atime: tempo do último acesso feito ao arquivo;
3. Mtime: tempo da última modificação;
4. Ctime: tempo de criação;
5. Tamanho: tamanho do arquivo regular;
6. ADS: número de *streams* de dados presentes no arquivo;
7. Tamanho total: tamanho total do arquivo, onde está incluido o tamanho das possíveis *alternate streams*.
8. Hash: hash criptgráfico SHA-1 do arquivo em questão;

Caso o arquivo não exista, o registro será composto apenas pelo nome do arquivo e pelos caracteres “ERR”.

B.2.1.2 Exemplo

Segue abaixo alguns exemplos de registro:

```
files.cfg;1034012980;1034012980;1014822243;77844;1;77,844;bf2c6d4ac532b73e0fc1caa6eb
554d83bab08833
audit.cfg;1034013145;1027004371;1017437051;628;1;628;8198c678b03c43622b00f1388a528f3
92eeae3d7
start.db;1034011615;1027006643;1017781619;4686;1;4,750;34614a8133934e32198b0049a2445
52c143d509b
C:\WinNT\TASKMAN.EXE;1026013179;944535600;1015248302;35600;1;35,600;ec71e91212ed7f42
62837ffde551a35e02ab08af
C:\WinNT\System32\Prounstl.exe;ERR
C:\WinNT\explorer.exe;1034013180;988988702;1015264069;242960;1;243,024;3e92e7dbf5922
9cba060c5fd2070af0e8c412500
C:\WinNT\hh.exe;1030013180;988988702;1015264069;26896;1;26,960;c6d0059f77eb477ed60d4
96efa31f38340546fbc
```

```
C:\WinNT\regedit.exe;1032813181;944535600;944535600;72464;1;72,464;122f4385ab55c9ec2
a3b4f6948bbfbd02dbbedb5
```

B.2.2 Start.db

O arquivo `start.db` é responsável por armazenar informações a respeito dos programas que são executados automaticamente quando a máquina inicia ou quando um usuário se autentica no console.

B.2.2.1 Dicionário de Dados

Existem quatro tipos de registro nesta base de dados, são eles:

- Tipo Diretório (D): armazena informações a respeito do diretório presente nos *profiles* dos usuários que armazenan atalhos para os programas que devem ser iniciados quando este se autentica no console da máquina (pasta *Startup*).

    1. Máquina: máquina monitorada;
    2. Tipo: tipo de registro (D);
    3. Diretório: caminho completo do diretório monitorado;
    4. Mtime: instante da última alteração;
    5. Ctime: instante de criação;
    6. Entradas: número de arquivos;

- Tipo Link (L): responsável por armazenar informações a respeito do estado dos atalhos contidos na pasta *Startup* e dos programas apontados por eles.

    1. Máquina: máquina monitorada
    2. Tipo: tipo de registro (L);
    3. Diretório: diretório ao qual pertence;
    4. Larquivo: nome do arquivo de atalho;
    5. LHash: hash criptográfico do arquivo de atalho;
    6. Programa: caminho completo até o programa apontado pelo atalho;
    7. Mtime: instante da última modificação no programa;
    8. Ctime: instante da criação do programa;
    9. ADS: número de *streams* de dados do programa;
    10. Tamanho Total: tamanho total do arquivo, incluido todas as ADSs;

11.Hash: hash criptográfico do programa;

- Tipo Chave (K): armazena informações sobre algumas chaves de registro, cujos valores indicam programas a serem executados quando a máquina se inicia, quando qualquer usuário se autentica no console, ou quando um usuário específico se autentica no console. O nome das chaves e suas respectivas descrições estão presentes na Seção 6.1.5.2.

  1. Máquina: máquina monitorada;
  2. Tipo: tipo de registro (K);
  3. Hierarquia: hierarquia (*hive*) à qual a chave pertence;
  4. Chave: caminho completo da chave;
  5. Valores: número de valores que possui;
  6. LWrite: instante da última alteração;

- Tipo Valor de Registro (V): responsável por armazenar dados sobre o estado dos valores presentes nas chaves de registro do ítem anterior, além de armazenar o estado dos programas apontados por elas.

  1. Máquina: máquina monitorada;
  2. Tipo: tipo de registro (V);
  3. Hierarquia: hierarquia (*hive*) à qual o valor pertence;
  4. Chave (Valor): nome do valor;
  5. Programa: programa referenciado pela chave;
  6. Mtime: instante da última modificação do programa;
  7. Ctime: instante de criação do programa;
  8. ADS: número de streams de dados do programa;
  9. Total Size: tamanho total do arquivo do programa;
  10. Hash: hash criptográfico do programa;

B.2.2.2 Exemplo

```
ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows NT/CurrentVersion/Winlogon/
UserInit;C:\WINNT\SYSTEM32\USERINIT.EXE;988988702;944535600;1;17,680;ec4b7c9ed408730
547428a7e06ba0847cf7f2585

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows NT/CurrentVersion/Windows/
Load;Undefined

ANAKIN;K;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run;11;Thu;7/18/2002;11-
00-15-401
```

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/QuickTime Task;C:\WINNT\SYSTEM32\QTTASK.EXE;1015438885;1015438885;1;28,672;222a694f6044bf99988f2804ea6b6eccda15cd14

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/WinampAgent;C:\PROGRAM FILES\WINAMP\WINAMPA.EXE;1001979720;1001979720;1;10,752;21baf13bcad02df06cf2b87e9164a7a369bf4714

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/InCD;C:\PROGRAM FILES\AHEAD\INCD\INCD.EXE;994181160;1016030569;1;753,664;a6e3a43b9f8cca7383d025017aa993f8a233e438

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/CMESys;C:\PROGRAM FILES\COMMON FILES\CMEII\CMESYS.EXE;1025727068;1014667915;1;86,016;cf8e95fc43c34a00e0dc43365bb98a32e6d8cd33

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/NvCplDaemon;C:\WINNT\system32\NVQTWK.DLL;987766920;1016800566;1;139,264;abe578de7fa3899a3d385f0f0740f98976cdea5f

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/TkBellExe;C:\PROGRAM FILES\COMMON FILES\REAL\UPDATE_OB\EVNTSVC.EXE;1020712682;1016728354;1;146,432;003850159d81946dfdd0026b59c2827f156d3cc3

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/MediaLoads Installer;C:\PROGRAM;;;;Undefined

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/New.net Startup;C:\PROGRA~1\NEWDOT~1\NEWDOT~1.DLL;1025289074;1025289076;1;167,936;e5821e07bffa3755712e0201293df10746ddab0f

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/POINTER;C:\MSSQL7\BINN\POINT32.EXE;;;;Undefined

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/SaveNow;C:\PROGRAM FILES\SAVENOW\SAVENOW.EXE;1021051454;1025705926;;f2304288ca51911ca09aa699f761aef293e281cd

ANAKIN;V;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/CreateCD;C:\PROGRA~1\ADAPTEC\EASYCD~1\CREATECD\CREATECD.EXE;1024953980;1024953941;1;262,208;19814146771840b3f1c1e497789c4b228ccbb30d

ANAKIN;K;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnce;0;Sat;6/29/2002;21-08-51-900

ANAKIN;K;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnceEx;0;Fri;3/15/2002;15-26-47-932

ANAKIN;K;LMachine;SOFTWARE/Microsoft/Windows/CurrentVersion/RunServices;Undefined

ANAKIN;K;CUser;SOFTWARE/Microsoft/Windows/CurrentVersion/Run;1;Sat;3/9/2002;15-00-29-952

ANAKIN;V;CUser;SOFTWARE/Microsoft/Windows/CurrentVersion/Run/internat.exe;C:\WINNT\system32\INTERNAT.EXE;944535600;944535600;1;20,752;08f1b7c33c0f5c3ed7aad0752bd63d96db1d4c2f

ANAKIN;K;CUser;SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnce;0;Mon;7/8/2002;11-14-53-694

ANAKIN;K;CUser;SOFTWARE/Microsoft/Windows/CurrentVersion/RunOnceEx;Undefined

```
ANAKIN;K;CUser;SOFTWARE/Microsoft/Windows/CurrentVersion/RunServices;Undefined

ANAKIN;D;C:\Documents and Settings;1020966178;1015248263;7

ANAKIN;D;C:\Documents and Settings\Administrator\start
menu\programs\startup;1015334059;1015263049;1

ANAKIN;D;C:\Documents and Settings\Administrator.ANAKIN\start
menu\programs\startup;1016559513;1016217480;2

ANAKIN;L;Administrator.ANAKIN;Command
Prompt.lnk;6fd9d2808a3aaeacf98517725c9e47a78173b3e6;C:\WINNT\SYSTEM32\CMD.EXE;988988
702;944535600;1;236,368;c653d5b88cdd2389ea27a54d2eefe78e741c8f7d

ANAKIN;D;C:\Documents and Settings\All Users\start
menu\programs\startup;1027000871;1015248297;5

ANAKIN;L;All Users;GetRight - Tray
Icon.lnk;f89ad10c98a0bd33b69ddc575b21873dede12d37;C:\PROGRAM
FILES\GETRIGHT\GETRIGHT.EXE;994693040;1015419068;1;2,227,264;508fdd2fa7bb1162bab86f2
d5e4d9156dd911828

ANAKIN;L;All Users;GStartup.lnk;41955a72e169821e62cb7f76f78adbb42bc1405c;C:\PROGRAM
FILES\COMMON
FILES\GMT\GMT.EXE;1025726806;1014667690;1;1,691,764;bc315caa913add5572e4c232c09ce7b6
1d30d7ab

ANAKIN;L;All Users;Microsoft
Office.lnk;9a9fbb25f2a51b486996f81e7425b24689b0dcaf;C:\PROGRAM FILES\MICROSOFT
OFFICE\OFFICE10\OSA.EXE;982065664;982065664;1;83,424;7309e21aae28d56bab2a93ee200ee2c
f32ce3742

ANAKIN;L;All Users;SpyCam.lnk;13b99620bcc8347b33c9b619bfa87558cbab2427;C:\PROGRAM
FILES\SPYCAM\SPYCAM.EXE;994121766;1024932765;1;409,664;57791c8775f7a9ef63ad9f9b2d335
195cc757673

ANAKIN;D;C:\Documents and Settings\Default User\start
menu\programs\startup;1015520230;1015248297;1

ANAKIN;D;C:\Documents and Settings\flavio\start
menu\programs\startup;1017440438;1016218245;1

ANAKIN;D;C:\Documents and Settings\fulano\start
menu\programs\startup;1015520230;1020966178;1
```

B.3 Utilização

O PFSAF possui uma interface interativa obtida através da execução do script `main.pl`, que fornece um menu de opções através do qual podem ser acessadas todas as funcionalidades do *framework* podem ser acessadas e cuja interface pode ser observada na Figura 6.6.

É possível executar também os *scripts* individualmente de forma não interativa através de linha de comando ou através do agendamento de tarefas no Task Scheduler (Figura B.1). O nome dos *scripts* a serem invocados e suas respectivas descrições podem ser obtidas na Seção 6.1.7.

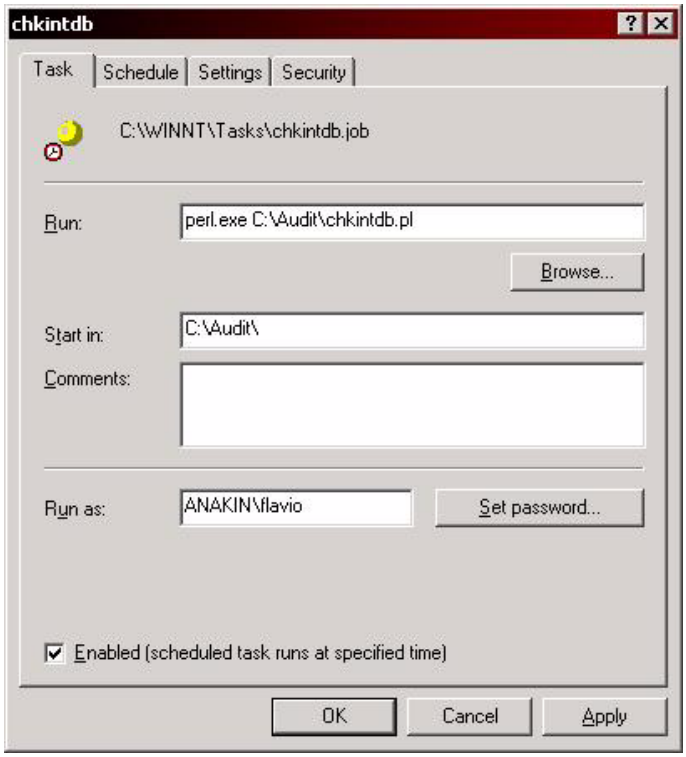


Figura B.1: Checagem de integridade agendada no Task Scheduler

Os procedimentos de checagem (`chkintdb.pl`, `chkstartdb.pl` e `chkaudit.pl`) produzem relatórios textuais indicando a ocorrêcia de alterações no estado dos objetos monitorados (Seção B.3.1). Os relatórios são armazenados em arquivos cujos nomes seguem o seguinte formato: `integrity-report_<Instante de Criação>.txt`.

B.3.1 Exemplo de relatório

```
------------------- Database Analysis Report -------------------
Base Directory: C:\Temp\Imagens\MyTripwire
Hash file     : hashs.db
Base SHA-1 Sum: 07c57e39b8c1b1f404e0bd894cffdb72b369d446
Date          : Tue Oct  8 21:24:46 2002
Report File   : integrity-report_1034123086.txt
----------------------------------------------------------------


File: files.cfg -> Ok!
```

```
File: audit.cfg -> Ok!
File: start.db -> Ok!
File: C:\WinNT\NOTEPAD.EXE -> Ok!
File: C:\WinNT\Speech\vcmd.exe -> Ok!
File: C:\WinNT\TASKMAN.EXE -> Ok!
File: C:\WinNT\delttsul.exe -> Ok!
File: C:\WinNT\explorer.exe -> Ok!
File: C:\WinNT\hh.exe -> Ok!
File: C:\WinNT\inf\unregmp2.exe  -> Ok!
File: C:\WinNT\msagent\agentsvr.exe -> Ok!
File: C:\WinNT\poledit.exe

Events Detected ->
Access Time      : Tue Oct  8 20:29:58 2002 <> Tue Oct  8 21:14:03 2002
Modification Time: Tue Dec  7 01:00:00 1999 <> Tue Oct  8 21:13:57 2002
Alternate Streams: 1       <> 2
Total Size       : 117,008 <> 118,107

File: C:\WinNT\regedit.exe -> Ok!
File: C:\WinNT\System32\AUTOCHK.EXE -> Ok!
File: C:\WinNT\System32\AUTOCONV.EXE -> Ok!
File: C:\WinNT\System32\CACLS.EXE -> Ok!
File: C:\WinNT\System32\CHKDSK.EXE  -> Ok!
```

B.4 Trechos de Código

Como já mencionado anteriormente, o PFSAF foi implementado utilizando a linguagem Perl que apesar de ter sido originalmente implementada para atuar em sistemas Unix, conta hoje com um grande número de bibliotecas específicas para o ambiente Windows.

A escolha da linguagem Perl em detrimento à linguagem VBScript, que é nativa do W2k, foi devido principalmente à facilidade do Perl em trabalhar com strings e a grande oferta de módulos específicos. Além disso o Perl é uma linguagem com licença GPL e pode garantir a portabilidade do PFSAF para outras plataformas Windows.

Nesta seção apresentamos alguns trechos de código relacionados principalmente à geração das bases de dados criadas pelo PFSAF.

B.4.1 Trecho do módulo de geração da base de integridade de arquivos (genintdb.pl)

...

```
#Checking cofiguration files...
foreach my $cfgfile (@config) {
 $adsnum = qx("adscount.pl $cfgfile");
 $base = hashsum($cfgfile);
 @macs = (lstat($cfgfile))[7..10];
 print "Hash $cfgfile: $base\n";
 print DB "$cfgfile;$macs[1];$macs[2];$macs[3];$macs[0];$adsnum;$base\n";
}

unless (open (POL, "$policy")) {
 print "Could not open policy file: $!\n";
 print "\n\n\nPress <RETURN> to continue....";
 $_ = <STDIN>;
 die "\n";
}

while (<POL>){
 $file = $_;
 chomp $file;
 if (-e $file) {
  $adsnum = qx("adscount.pl $file");
  $base = hashsum($file);
  @macs = (lstat($file))[7..10];
  print "Hash $file: $base\n";
  print DB "$file;$macs[1];$macs[2];$macs[3];$macs[0];$adsnum;$base\n"
 }
 else {
       print DB "$file;ERR\n";
       $err = 1;
       print "Can't open $file...\n";
 }
}
close(POL);
close(DB);
$base = hashsum($hashs);
open (SUM,"> $hashs.sum") || die "Could not open sum file: $!\n";
print SUM "$hashs\t$base";
close(SUM);
if ($err == 1) { print "\n\nThere were files NOT processed in the policy!!!"; }
print "\n\n\nPress <RETURN> to continue....";
$_ = <STDIN>;

sub hashsum {
  my ($file) = @_;
  open (FILE, "$file") || die "Can't open $file: $!\n";
  binmode(FILE);
  $digest = Digest::SHA1->new->addfile(*FILE)->hexdigest;
  close (FILE);
```

```
  return $digest;
}
```

B.4.2 Trechos do módulo de geração da base de integridade de programas executados automáticamente (genstartdb.pl)

...

```
if (open(ADTPLY,"$policy")) {
 while (<ADTPLY>) {
  $line = $_;
  chomp $line;
  next if ($line =~ "#" || $line eq "");
  $server = (split(/\(/,$line))[0];
  print "Machine: $server\n\n";
  $aux = (split(/\(/,$line))[1];
  $aux = (split(/\)/,$aux))[0];
  my @audit = split(/,/,$aux);
  die "Syntax error on $policy: line $.!\n" if ($#audit != 8);

  $server = "\U$server";
  $me = "\U$me";
  if (&sysstart($server,$me)) {
  &userstart($server,$me);
  &chkdirs($server,$me);
  }
  else {
   print DB "$server;S;Error\n";
  }
  print "------------------------------------------------------------\n";
 }
}
else {
      die "Can't open $policy: $!\n";
}
close($policy);
close(DB);

sub addkey {
 my ($server,$hive,$rpath) = @_;
 my $timestr = `keytime.pl $server \"$hive/$rpath\"`;
 print DB "$server;K;$hive;$rpath;$timestr\n";
}

#########################################################
#
#Analyse registry keys
#

sub getTrojanKeys {
 my ($server,$me,$hive,$remote) = @_;
 my ($rpath) = 'SOFTWARE/Microsoft/Windows/CurrentVersion';
 my (@keys) = ('Run','RunOnce','RunOnceEx','RunServices');
 foreach my $k (@keys) {
```

```
  print "Checking Key: $hive/$rpath/$k\n\n";
  &addkey($server,$hive,"$rpath/$k");
  my $key = $remote->{"$rpath/$k"};
  if (defined $key) {
   my @vals = $key->ValueNames;
   if($#vals != -1) {
    foreach my $val (@vals) {
     my $data = $key->{$val};
     $data = "NotFound" unless($data);
     my @files = &namesynt($server,$me,$data);
     foreach my $file (@files) {
      my $base = hashsum($file);
      my $adsnum = `adscount.pl \"$file\"`;
      my ($m,$c) = (lstat($file))[9..10];
      if ($server ne $me) {
       my @temp = split(/\\/,$file);
       my $drive = $temp[3];
       $drive =~ s/\$/:\\/g;
       $file = join("\\",@temp[4..$#temp]);
       $file = $drive.$file;
      }
      print "> $file -> $base\n";
      print DB "$server;V;$hive;$rpath/$k/$val;$file;$m;$c;$adsnum;$base\n";
     }
    }
   }
  }
 }
}


sub hashsum {
 my ($file) = @_;
 my $hkey = $senha;
 if (open (FILE, "$file")) {
  binmode(FILE);
  my $digest = Digest::HMAC_SHA1->new($hkey)->addfile(*FILE)->hexdigest;
  close (FILE);
  return $digest;
 }
 else {
   print "Can't open $file: $!\n";
   return("Undefined");
 }
}

sub recognize {
 my ($c) = @_;
 return(1) if ((($c eq "RUNDLL") || ($c eq "RUNDLL.EXE")) || (($c eq "RUNDLL32") ||
($c eq "RUNDLL32.EXE")));
 return(1) if ($c eq "START");
 return(1) if (($c eq "CMD") || ($c eq "CMD.EXE"));
 return(1) if ((($c eq "WSCRIPT") || ($c eq "CSCRIPT")) || (($c eq "WSCRIPT.EXE") ||
($c eq "CSCRIPT.EXE")));
 return(0);
```

```
}

sub namesynt {
 my ($server,$me,$brute) = @_;
 my ($aux,$final,$drive,$startdir,@temp,@files);
 my (@ext,@spath);
 my $e;

 if (($brute =~ /^\"/ ) && ($brute =~ /\"$/ )){
  @temp = split(/\"/,$brute);
  $brute = join("\"",@temp[1..$#temp]);
 }
 $aux = `parser.pl \"$brute\"`;              #parse line...
 my @targets = split(/;/,$aux);
 foreach my $target (@targets) {
  @temp = split(/\\/,$target);
  next if (((!$#temp) && ($target eq $targets[0])) && recognize($temp[$#temp]));
  $aux = $temp[$#temp];

  if (!$#temp) {
   if ($aux =~ /./) {
    @spath = &getspath($server);
    foreach my $sp (@spath) {
     ($drive,$sp) = split(/:/,$sp);
     if ($server eq $me){
      $startdir = "$drive:";
     }
     else {
      $startdir = "\\\\$server\\$drive\$";
     }
     $sp = $sp."\\" if !($sp =~ /\\$/);
     $final = $startdir.$sp.$target;
     last if (-e $final);
    }
   }
   else {
    @ext = &getext($server);
    @spath = &getspath($server);
    foreach my $sp (@spath) {
     ($drive,$sp) = split(/:/,$sp);
     if ($server eq $me){
      $startdir = "$drive:";
     }
     else {
      $startdir = "\\\\$server\\$drive\$";
     }
     $sp = $sp."\\" if !($sp =~ /\\$/);
     foreach $e (@ext) {
      $final = $sp.$target.$e;
      last if (-e $startdir.$final);
     }
     last if (-e $startdir.$final);
    }
   }
  }
```

```
  else {
   ($drive,$aux) = split(/:/,$target);
   if ($server eq $me) {
    $startdir = "$drive:";
   }
   else { $startdir = "\\\\$server\\$drive\$"; }
   $final = $startdir.$aux;
  }
  push(@files,$final);
 }
 return(@files);
}

sub chkdirs {
 my ($server,$me) = @_;

 print "\n- Checking Startup Directories\n\n";
 my ($startdir,$drivechar,$xdir,$drive,@temp);
 $drivechar = &getsysdrivel($server);
 if ($server eq $me) {
  $startdir = "$drivechar:";
 }
 else {
  $startdir = "\\\\$server\\$drivechar\$";
 }
 my $start = $startdir."\\Documents and Settings";
 my $startup = "\\start menu\\programs\\startup";
 my ($dir,$err);
 if (-e $start && -d $start) {
  opendir(ST,"$start");
  my @files = readdir(ST);
  rewinddir(ST);
  my ($m,$c) = (lstat($start))[9..10];
  if ($server ne $me) {
   @temp = split(/\\/,$start);
   $drive = $temp[3];
   $drive =~ s/\$/:\\/g;
   $xdir = join("\\",@temp[4..$#temp]);
   $xdir = $drive.$xdir;
  }
  else {
   $xdir = $start;
  }
  print DB "$server;D;$xdir;$m;$c;$#files\n";
  foreach $dir (sort readdir(ST)) {
   next if ($dir eq "." || $dir eq "..");
   my $dir2 = $start."\\".$dir;
   if (-e $dir2 && -d $dir2) {
    my $newdir = "$start\\".$dir."$startup";
    if (-e $newdir && -d $newdir) {
     opendir(SUP,$newdir);
     @files = readdir(SUP);
     closedir(SUP);
     ($m,$c) = (lstat($newdir))[9..10];
     if ($server ne $me) {
```

```
      @temp = split(/\\/,$newdir);
      $drive = $temp[3];
      $drive =~ s/\$/:\\/g;
      $xdir = join("\\",@temp[4..$#temp]);
      $xdir = $drive.$xdir;
     }
     else { $xdir = $newdir; }
     print DB "$server;D;$xdir;$m;$c;$#files\n";
     print "Cheking: $xdir\n";
     if (@files) {
      foreach my $file (@files) {
       next if ($file eq "." || $file eq "..");
       if ($file =~ m/lnk$/) {
        my $shortcut = Win32::Shortcut->new($newdir."\\".$file);
        my $basel = hashsum($newdir."\\".$file);
        if ($shortcut) {
         my $data = $shortcut->{Path};
         my @xfiles = &namesynt($server,$me,$data);
         foreach my $xfile (@xfiles) {
          my $base = hashsum($xfile);
          my $adsnum = `adscount.pl \"$xfile\"`;
          ($m,$c) = (lstat($xfile))[9..10];
          if ($server ne $me) {
           my @temp = split(/\\/,$xfile);
           my $drive = $temp[3];
           $drive =~ s/\$/:\\/g;
           $xfile = join("\\",@temp[4..$#temp]);
           $xfile = $drive.$xfile;
          }
          print DB "$server;L;$dir;$file;$basel;$xfile;$m;$c;$adsnum;$base\n";
          print "> $xfile -> $base\n";
         }
        }
        else {
         print "Error with Shortcut: ".Win32::FormatMessage Win32::GetLastError."\n";
         print DB "$server;L;$dir;$file;Undefined\n";
        }
       }
       else {
        print "$dir:$file\n";
       }
      }
     }
    }
    else {
     print "$newdir does not exist or is not a directory.\n";
    }
   }
  else {
   print "$dir2 does not exist or is not a directory.\n";
  }
}
closedir(ST);
}
else {
```

```
  print "$start does not exist or is not a directory.\n";
 }
}

sub sysstart {
 my ($server,$me) = @_;

 my %rkeys = ("UserInit" => "SOFTWARE/Microsoft/Windows NT/CurrentVersion/Winlogon/
UserInit",
              "Load" => "SOFTWARE/Microsoft/Windows NT/CurrentVersion/Windows/Load");
 my ($file);
 my $remote = &conn_hive($server,'LMachine');
 return(0) unless ($remote);
 foreach my $k (keys %rkeys) {
  my $reg = $rkeys{$k};
  my $data = $remote->{$reg};
  my ($drive,@files,@temp);
  if (defined $data) {
   @files = &namesynt($server,$me,$data);
   foreach $file (@files) {
    my $base = hashsum($file);
    my $adsnum = `adscount.pl \"$file\"`;
    my ($m,$c) = (lstat($file))[9..10];
    if ($server ne $me) {
     @temp = split(/\\/,$file);
     $drive = $temp[3];
     $drive =~ s/\$/:\\/g;
     $file = join("\\",@temp[4..$#temp]);
     $file = $drive.$file;
    }
    print DB "$server;V;LMachine;$reg;$file;$m;$c;$adsnum;$base\n";
#server;type;hive;path;program;mtime;ctime;nads;sizet;hash
   }
  }
  else { print DB "$server;V;LMachine;$reg;Undefined\n"; }
 }
 return(1);
}

sub userstart {
 my ($server,$me) = @_;

 my %hives = ("LMachine" => "HKLM",
              "CUser" => "HKCU");
 my ($remote);

 foreach my $hive (keys %hives) {
  $remote = &conn_hive($server,$hive);
  next unless ($remote);
  &getTrojanKeys($server,$me,$hive,$remote);
 }
}

sub getsysdrivel {
 my ($server) = @_;
```

```
 my $rootdir = &readreg($server,'LMachine','SOFTWARE/Microsoft/Windows NT/
CurrentVersion/SystemRoot');
 my $drivechar = (split(/:/,$rootdir))[0];
 return($drivechar);                        #return the drive letter
}

sub getext {
 my ($server,$remote) = @_;
 my $data = &readreg($server,'LMachine','SYSTEM/CurrentControlSet/Control/Session
Manager/Environment/PATHEXT');
 my @extensions = split(/;/,$data);
 return (@extensions); #return extensions automaticaly executed
}

sub getspath {
 my ($server) = @_;
 my $data = &readreg($server,'LMachine','SYSTEM/CurrentControlSet/Control/Session
Manager/Environment/Path');
 my @dirs = split(/;/,$data);
 foreach my $i (0..$#dirs) {
  $dirs[$i] = &expand($server,$dirs[$i]);
 }
 return(@dirs);                             #return list of path dirs
}

sub readreg{
 my ($machine,$hive,$path) = @_;
 my ($remote,$result);
 if ($remote = $Registry->{"//$machine/$hive"}) {
  $result = $remote->{$path};
 }
 else {
  print "Can't connect with: //$machine/$hive\n";
  return;
 }
 print "Can't read: $hive/$path\n" if (!$result);
 return($result);                          #return key contents..
}

sub conn_hive {
 my ($machine,$hive) = @_;
 my ($remote);
 if ($remote = $Registry->{"//$machine/$hive"}) {
  return($remote);
 }
 else {
  print "Can't connect with: //$machine/$hive\n";
  return;
 }
}

sub expand {
 my ($server,$path) = @_;
 my $systemroot = &readreg($server,'LMachine','SOFTWARE/Microsoft/Windows NT/
CurrentVersion/SystemRoot');
```

```
 my $ProgramFilesDir = &readreg($server,'LMachine','SOFTWARE/Microsoft/Windows/
CurrentVersion/ProgramFilesDir');
 $path =~ s/%systemroot%/$systemroot/gi;
 $path =~ s/%ProgramFilesDir%/$ProgramFilesDir/gi;
 $path =~ s/%ProgramFilesPath%/$ProgramFilesDir/gi;
 return ($path);
}
```

B.4.3 Trechos do Módulo para geração e checagem da base de autitoria (setaudit.pl e chkaudit.pl)

...

```
if (open(ADTPLY,"$policy")) {
 while (<ADTPLY>) {
  $line = $_;
  chomp $line;
  next if ($line =~ "#" || $line eq "");
  $machine = (split(/\(/,$line))[0];
  $aux = (split(/\(/,$line))[1];
  $aux = (split(/\)/,$aux))[0];
  @audit = split(/,/,$aux);
  if ($#audit != 8) { die "Syntax error on $policy: line $.!\n";}
   &setting($machine,@audit);
 }
}
else {
      die "Can't open $policy: $!\n";
}
close($policy);
($dirty == 0) ? print "\nAudit Policy successfully configured!!!\n":print "\nThere
were errors during the configuration!!!\n";
print "\n\nPress <RETURN> to continue...";
$_ = <STDIN>;


sub setting {

 my ($server,@audit) = @_;

 my (%info,@options);
 my @macros =
(&POLICY_AUDIT_EVENT_NONE,&POLICY_AUDIT_EVENT_SUCCESS,&POLICY_AUDIT_EVENT_FAILURE,
                &POLICY_AUDIT_EVENT_SUCCESS |
&POLICY_AUDIT_EVENT_FAILURE,&POLICY_AUDIT_EVENT_UNCHANGED);

 print "\n$server -> @audit";


 $options[AuditCategoryDetailedTracking] = $macros[$audit[0]];
 $options[AuditCategoryAccountLogon] = $macros[$audit[1]];
 $options[AuditCategorySystem] = $macros[$audit[2]];
 $options[AuditCategoryAccountManagement] = $macros[$audit[3]];
 $options[AuditCategoryDirectoryServiceAccess] = $macros[$audit[4]];
```

```
 $options[AuditCategoryObjectAccess] = $macros[$audit[5]];
 $options[AuditCategoryPolicyChange] = $macros[$audit[6]];
 $options[AuditCategoryLogon] = $macros[$audit[7]];
 $options[AuditCategoryPrivilegeUse] = $macros[$audit[8]];

 %info = ('auditingmode' => 1,                  # turn on auditing
          'eventauditingoptions' => \@options );

 if(!Win32::Lanman::LsaSetAuditEventsPolicy("\\\\$server", \%info)){
  print "\tFAIL!\n";
  print "Error: ";
  print Win32::FormatMessage(Win32::Lanman::GetLastError());
  $dirty = 1;
 }
 else { print "\tOK!\n";}
}


############################
#
#Module to check the audity policy configuration state
#
sub auditpol {
 my ($server,@audit) = @_;
 my (%info);
 my ($dirty) = 0;
 my (%hash) = ("AuditCategoryDetailedTracking" => "Process Tracking",
               "AuditCategoryAccountLogon" => "Account Logon",
               "AuditCategorySystem" => "Restart and Shutdown System",
               "AuditCategoryAccountManagement" => "User/Group Management",
               "AuditCategoryDirectoryServiceAccess" => "Dir. Service Access",
               "AuditCategoryObjectAccess" => "File and Object Access",
               "AuditCategoryPolicyChange" => "Security Policy Changes",
               "AuditCategoryLogon" => "Logon and Logoff",
               "AuditCategoryPrivilegeUse" => "Use of User Rights");

 my (%config) = ("AuditCategoryDetailedTracking" => $audit[0],
                 "AuditCategoryAccountLogon" => $audit[1],
                 "AuditCategorySystem" => $audit[2],
                 "AuditCategoryAccountManagement" => $audit[3],
                 "AuditCategoryDirectoryServiceAccess" => $audit[4],
                 "AuditCategoryObjectAccess" => $audit[5],
                 "AuditCategoryPolicyChange" => $audit[6],
                 "AuditCategoryLogon" => $audit[7],
                 "AuditCategoryPrivilegeUse" => $audit[8]);

 my (%settings) = (0 => "None",
                   1 => "Success",
                   2 => "Failure",
                   3 => "Success and Failure");


 print "\n\nSever: $server";
 if (Win32::Lanman::LsaQueryAuditEventsPolicy("\\\\$server",\%info)) {
  my $audit = $info{auditingmode};
```

```
  if (!$audit) {
   print "\n\n  Events Detected:\n";
   print "\tError: Auditing is NOT enabled!!!\n";
   $fails = 1;
  }
  else {
   my $options = $info{eventauditingoptions};
   foreach my $key (keys %hash) {
    if ($$options[&$key] != $config{$key}){
     if ($dirty == 0){
      print "\n\n  Events Detected:\n";
      $dirty = 1;
      $fails = 1;
     }
     print "\t$hash{$key} -> $settings{$$options[&$key]} <>
$settings{$config{$key}}\n";
    }
   }
  }
  print " -> OK!" if ($dirty == 0);
 }
 else {
  print "\n\n  Events Detected:\n";
  print "\tError: ";
  print Win32::FormatMessage(Win32::Lanman::GetLastError());
  $fails = 1
 }
}
```

Bibliografia

[1] AUGUSTO, Alessandro; de GEUS, Paulo L.; GUIMARÃES, Célio C.; *Administration of Large Windows NT Network with DoIt4Me*; The 10th International Conference on System Administration, Networking and Security; Baltimore, MD, EUA; maio 2001;

[2] BREZINSKI, Dominique; *Building a Forensic Toolkit That Will Protect You From Evil Influences*; The Black Hat Briefings '99; Las Vegas, NV, EUA; julho 1999;

[3] BROWNLEE, N.; GUTTMAN, E.; *Expectations for Computer Security Incident Response*; Internet Engineering Task Force; Request for Comments 2350; 1998;

[4] CAIS; *Site do Centro de Atendimento a Incidentes de Segurança*; Disponível em: <http://www.rnp.br/cais/>; Acesso em: 28/10/2002;

[5] CARISSIMI, Leonardo; *Inimigo na Trincheira*; Modulo Security Solutions S/A; Disponível em: <http://www.modulo.com.br/empresa/noticias/artigo_entrevista/a-trincheira.htm>; Acesso em: 28/05/2002;

[6] CARPENTER, Jeffrey J.; *Welcome to The Big City*; ;login: The Magazine of USENIX & SAGE; novembro 1999;

[7] CARRIER, Brian; *An Investigator's Guide to File System Internals*; Proc. of the 14th Annual FIRST Computer Security Incident Handling Conference; Hawaii, EUA; junho 2002;

[8] CARVEY, Harlan; *System Security Administration for NT*; LISA-NT -- The 3rd Large Installation System Administration of Windows NT Conference; Seattle, Washington - EUA; 2000;

[9] CARVEY, Harlan; *Carvdawg's Perl Page*; Disponível em: <http://patriot.net/~carvdawg/perl.html>; Acesso em: 29/08/2002;

[10] CASEY, Eoghan; *Handbook of Computer Crime Investigation - Forensic Tools an Technology*; Academic Press; 1ª Edição; 2001;

[11] CERT/CC, Computer Emergency Response Team/Cordination Center; *Incident Reporting Guidelines;* Carnegie Mellon Software Engineering Institute; julho 2001; Disponível em: <http://www.cert.org/tech_tips/incident_reporting.html>; Acesso em: 27/05/2002;

[12] CHAPMAN, D. Brent; ZWICKY, Elizabeth D.; COOPER Simon; *Building Internet Firewalls*; O'Reilly Associates; 2ª edição, 2000;

[13] CNN; *O. J. Simpson Trial*; Disponível em: <http://www.cnn.com/US/OJ/index.html>; Acesso em: 01/08/2002;

[14] DEFENSE, US Department of; ENERGY, US Department of; US NUCLEAR REGULATORY COMMISSION; US CENTRAL INTELLIGENCE AGENCY; *National Industrial Security ProgramOperating Manual (NISPOM)*; DoD 5220.22-M; Disponível em: <http://www.dss.mil/isec/nispom.htm>; Acesso em: 05/08/2002;

[15] DEFENSE, US Department of; *Clearing and Sanitization Matrix*; Disponível em: <http://www.dss.mil/infoas/clearing_and_sanitization_matrix.doc>; Acesso em: 05/08/2002;

[16] DITTRICH, Dave; *Basic Steps in Forensic Analysis of Unix Systems*; Disponível em: <http://staff.washington.edu/dittrich/misc/forensics/>; Acesso em: 04/09/2002;

[17] ESPINDULA, Alberi; *A Função Pericial do Estado*; Associação Nacional dos Peritos Criminais Federais; Brasília - DF; 2000;

[18] FARMER, Dan; *What are MACTimes?*; Dr Dobb's Journal; outubro 2000; Disponível em: <http://www.ddj.com/documents/s=880/ddj0010f/0010f.htm>; Acesso em: 03/08/2002;

[19] FARMER, Dan; VENEMA, Wietse; *Forensic Computer Analysis: An Introduction*; Dr. Dobb's Journal; setembro de 2000; Disponível em: <http://www.ddj.com/documents/s=881/ddj0009f/0009f.htm>; Acesso em: 18/06/2002;

[20] FEDERAL Bureau of Investigation Home Page; Disponível em: <http://www.fbi.gov>; Acesso em: 29/08/2002;

[21] GLASER, J. D.; *Auditing NT - Catching Greg Hoglund*; The Black Hat Briefings '99; Las Vegas, NV, EUA; julho 1999;

[22] HONEYNET Project; Disponível em: <http://project.honeynet.org/>; Acesso em: 29/08/2002;

[23] HTCIA International; *High Technology Crime Investigation*; Disponível em: <http://www.htcia.org>; Acesso em: 29/08/2002;

[24] KARSPERSKY, Eugene; ZENKIN, Denis; *NTFS Alternate Data Streams*; Windows 2000 Magazine; Storage Admin; spring 2001; Disponível em: <http://www.storageadmin.com/Articles/Index.cfm?ArticleID=19878>; Acesso em: 03/08/2002;

[25] KOSSAKOWSKI, Klaus-Peter; ALLEN, Julia; ALBERTS, Christopher; COHEN, Cory; FORD, Gary; FRASER, Barbara; HAYES, Eric; KOCHMAR, John; KONDA, Suresh; WILSON, William; *Respondig to Intrusions*; Carnegie Mellon Software Engineering Institute; 1ª Edição; 1999;

[26] LEE, Henry; PALMBACH, Timothy; MILLER, Marilyn; *Henry Lee's Crime Scene Handbook*; Academic Press; julho 2001;

[27] LEE, Rob; *Incident Response, Computer Forensic Analysis, and Electronic Investigations*; Disponível em: <http://www.incident-response.org>; Acesso em: 02/09/2002;

[28] MANDIA, Kevin; PROSISE, Chris; *Incident Response: Ivestigating Computer Crime*; Osborne/McGraw-Hill; 1ª Edição; junho 2001;

[29] MENEZES, Alfred J.; van OORSCHOT, Paul C.; VANSTONE, Scott A.; *Handbook of Applied Cryptography*; CRC Press; 1996;

[30] McMILLAN, Jim; *Importance of a Standard Methodology in Computer Forensics*; Information Security Reading Room; SANS Institute; Disponível em: <http://www.sans.org/infosec-FAQ/incident/methodology.htm>; Acesso em: 15/01/2002;

[31] MICROSOFT, Corp; *Description of RunOnceEx and RunEx Registry Keys [Q232487]*; Microsoft TechNet; Microsoft Corporation; Julho 2000;

[32] MICROSOFT, Corp; *HOW TO: Backup, Edit, and Restore the Registry in Windows 2000 [*Q322755*]*; Microsoft Technet; Maio 2002;

[33] MICROSOFT, Corp; *How to Use Dumpchk.exe to Check a Memory Dump File [Q156280]*; Microsoft Technet; Setembro 1996;

[34] MICROSOFT, Corp.; *MCSE Training Kit – Microsoft Windows 2000 Active Directory Services*; Microsoft Press; 2000;

[35] MICROSOFT, Corp.; *Securing Windows® 2000 Network Resources*; Microsoft TechNet; Microsoft Corporation; Julho 2000;

[36] MICROSOFT, Corp; *The Windows 95 Rundll and Rundll32 Interface [Q164787]*; Microsoft TechNet; Microsoft Corporation; Julho 2000;

[37] MICROSOFT, Corp; *Windows 2000 Memory Dump Options Overview [Q254649]*; Microsoft Technet; Fevereiro 2000;

[38] MICROSOFT, Corp; *Windows 2000 Professional Resource Kit*; Microsoft Technet; Microsoft Corporation; Janeiro 2000;

[39] MICROSOFT, Corp; *Windows Feature Allows a Memory.dmp File to Be Generated with Keyboard [Q244139]*; Microsoft Technet; Outubro 1999;

[40] MSNBC; *Judge OKs FBI hack of Russian computers*; Disponível em: <http://zdnet.com.com/2100-11-529917.html?legacy=zdnn>; Acesso em: 01/08/2002;

[41] NOBLETT, Michael G.; POLLITT, Mark M.; PRESLEY, Lawrence A.; *Recovering and Examining Computer Forensic Evidence*; Forense Science Communications; Federal Bureau of Investigation; Outubro 2000, Vol. 2 N. 4;

[42] NBSO; *Site do NIC BR Security Office*; Disponível em: <http://www. nic.br/nbso.html>; Acesso em: 29/08/2002;

[43] NOBLETT, Michael G.; *Report of the Federal Bureau of Investigation on development of forensic tools and examinations for data recovery from computer evidence*; Proceedings of the 11th INTERPOL Forensic Sciense Symposium; 1995;

[44] OLIVEIRA, Flávio de Souza; REIS, Marcelo Abdalla dos; CARDOSO, Célio Guimarães; GEUS, Paulo Lício; *Forense Computacional: Aspectos Legais e de Padronização*; 9º Simpósio de Computação Tolerante a Falhas - I Workshop em Segurança de Sistemas Computacionais (WSeg'2001); p. 80-85; Florianópolis, Santa Catarina, Brasil; março 2001;

[45] OLIVEIRA, Flávio de Souza; CARDOSO, Célio Guimarães; GEUS, Paulo Lício; *Metodologias de Análise Forense para Ambientes Baseados em NTFS*; III Simpósio de Segurança em Informática (SSI'2001); p. 83-90; São José dos Campos, São Paulo, Brasil; outubro 2001;

[46] OLIVEIRA, Flávio de Souza; CARDOSO, Célio Guimarães; GEUS, Paulo Lício; *Resposta a Incidentes para Ambientes Corporativos Baseados em Windows*; 20º Simpósio Brasileiro de Redes de Computadores - II Workshop em Segurança de Sistem as Computacionais (WSeg'2002); p. 128-135; Búzios, Rio de Janeiro, Brasil; maio 2002;

[47] OLIVEIRA, Flávio de Souza; CARDOSO, Célio Guimarães; GEUS, Paulo Lício; *Um Framework para Preparação de Redes Windows 2000 para Futuras Análises Forenses*; IV Simpósio de Segurança em Informática (SSI'2002); novembro 2002;

[48] OLIVEIRA, Flávio de Souza; CARDOSO, Célio Guimarães; GEUS, Paulo Lício; *Pre-Forensic Setup Automation for Windows 2000*; IASTED International Conference on Communications and Computer Networks (CCN'02); Massachusetts Institute of Technology (MIT); Cambridge; Massachusetts; EUA; novembro 2002;

[49] PETHIA, R.; CROCKER, S.; FRASER, B.; *Guidelines for the Secure Operation of the Internet*; Internet Engineering Task Force; Request for Comments 1281; 1991;

[50] REIS, Marcelo A.; de GEUS, Paulo L.; *Standardization of Computer Forensic Protocols and Procedures*; Proc. of the 14th Annual FIRST Computer Security Incident Handling Conference; Hawaii, EUA; June 2002;

[51] RUSSEL, Charlie; CRAWFORD, Sharon; *Microsoft® Windows® 2000 Server Administrator's Companion*; Microsoft Press; 2000;

[52] RUSSINOVICH, Mark; *Exploring NTFS On-disk Structures*; Windows 2000 Magazine; Focus; novembro 2000;

[53] RUSSINOVICH, Mark; *Inside Win2K NTFS, Part 2*; Windows 2000 Magazine; Focus; winter 2000; Disponível em: <http://www.win2000mag.com/Articles/Index.cfm?ArticleID=15900>; Acesso em 03/08/2002;

[54] SMIT, Thomas A.; *An In-depth Look at W32.Winux and W2K.Stream and the Ever Growing "Proof of Concept" Virus*; Information Security Reading Room;
Disponível em: <http://www.sans.org/infosecFAQ/malicious/w2kwinux.htm>; Acesso em: 12/11/2001;

[55] SPITZNER, Lance; *Know Your Enemy: A Forensics Analysis*; Honeynet Project; Disponível em: <http://project.honeynet.org/papers/forensics/>; Acesso em: 04/09/2002;

[56] SPITZNER, Lance; *Know Your Enemy II: Tracking the blackhat's moves*; Honeynet Project; Disponível em: <http://project.honeynet.org/papers/enemy2/>; Acesso em: 04/09/2002;

[57] SWGDE, Scientific Working Group on Digital Evidence; IOCE, International Organization on Digital Evidence; *Digital Evidence: Standards and Priciples*; Forense Science Communications; Federal Bureau of Investigation; Abril 2000, Vol. 2 N. 2;

[58] SWGMAT, Scientific Working Group on Materials Analysis; *Trace Evidence Recovery Guidelines*; Forense Science Communications; Federal Bureau of Investigation; Outubro 1999, Vol. 1 N. 3;

[59] THORTON, J.; *The general assumptions and rationale of forensic identification*; Modern Scientific Evidence: The Law and Science of Expert Testimony; West Publishing Co.; Volume 2; 1997;

[60] TRIPWIRE Open Source's Home Page; Disponível em: <http://www.tripwire.org>; Acesso em: 29/08/2002;

[61] VENEMA, Wietse; *Strangers In the Night - Findig the purpose of an unknown program*; Dr. Dobb's Journal; novembro 2000; Disponível em: <http://www.ddj.com/documents/s=879/ddj0011g/0011g.htm>; Acesso em: 14/08/2002;

[62] VENEMA, Wietse; *File Recovery Techniques*; Dr. Dobb's Journal; dezembro 2000; Disponível em: <http://www.ddj.com/documents/s=878/ddj0012h/0012h.htm>; Acesso em: 01/08/2002;

[63] VENEMA, Wietse; *Wietse Venema Website*; Disponível em: <http://www.porcupine.org>; Acesso em: 29/08/2002;

[64] W2K.STREAM; *Página criada pelos desenvolvedores do W2K.Stream*; Disponível em: <http://viry.bonusweb.cz/kniha_o_virech/ntfs.html>; Acesso em: 08/10/2001;

[65] WYK, Kenneth R. van; FORNO, Richard; *Incident Response*; O'Reilly*;* 1ª Edição; 2001;

[66] WEST-BROWN, Moira J.; STIKVOORT, Don; KOSSAKOWSKI, Klaus-Peter; *Handbook for Computer Security Incident Response Teams (CSIRTs)*; Carnegie Mellon Software Engineering Institute; 1ª Edição; 1998;